Natural for Ajax

Natural Page Layout

Version 9.3.4

October 2026

ADABAS & NATURAL

This document applies to Natural for Ajax Version 9.3.4 and all subsequent releases.

Specifications contained herein are subject to change and these changes will be reported in subsequent release notes or new editions.

Copyright © 2009-2026 Software GmbH, Darmstadt, Germany and/or its subsidiaries and/or its affiliates and/or their licensors.

The name Software AG and all Software GmbH product names are either trademarks or registered trademarks of Software GmbH and/or its subsidiaries and/or its affiliates and/or their licensors. Other company and product names mentioned herein may be trademarks of their respective owners.

Detailed information on trademarks and patents owned by Software GmbH and/or its subsidiaries is located at https://softwareag.com/licenses.

Use of this software is subject to adherence to Software GmbH's licensing conditions and terms. These terms are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

This software may include portions of third-party products. For third-party copyright notices, license terms, additional rights or restrictions, please refer to "License Texts, Copyright Notices and Disclaimers of Third-Party Products". For certain specific third-party license restrictions, please refer to section E of the Legal Notices available under "License Terms and Conditions for Use of Software GmbH Products / Copyright and Trademark Notices of Software GmbH Products". These documents are part of the product documentation, located at https://softwareag.com/licenses and/or in the root installation directory of the licensed product(s).

Use, reproduction, transfer, publication or disclosure is prohibited except as specifically provided for in your License Agreement with Software GmbH.

Document ID: ONE-NATNJX-PAGELAYOUT-NATURAL-934-20261004

Table of Contents

Natural Page Layout xv
1 About this Documentation 1
Document Conventions 2
Online Information and Support 2
Data Protection 3
I Typical Page Layout 5
2 NATPAGE 7
Properties 8
3 TITLEBAR 17
Properties 18
4 HEADER 23
Properties 24
5 PAGEBODY 25
Properties 26
6 STATUSBAR 29
General Information 30
Example 30
Properties 31
II Working with Containers 33
7 Positioning of Controls inside a Container 35
Row Types - TR and ITR 36
Some More Details on ITR 37
ITR in Google Chrome and Edge Chromium 38
TR Properties 39
ITR Properties 40
8 Defining the Width of Controls inside a Container 43
Controlling the Width of Controls 44
HDIST and VDIST Controls 46
HDIST Properties 48
VDIST Properties 48
rowspan and colspan Definitions 49
CELLSPAN Control 50
CELLSPAN Properties 51
Rules for Positioning Controls inside Containers 53
9 Vertical Sizing of Containers and Controls 55
Vertical Pixel Sizing 56
Vertical Percentage Sizing 57
Finishing the Example 59
10 Overview of Different Containers 61
Different Kind of Containers 62
Row Containers 62
Column Containers 63
Row and Column Containers in Combination 64

Nesting Containers 65
11 ROWAREA and COLAREA 67
ROWAREA Properties 68
COLAREA Properties 75
12 ROWAREAWITHHEADER 81
Simple Example 82
Right-to-left (RTL) Mode 83
ROWAREAWITHHEADER Properties 83
ROWAREAHEADER Properties 86
ROWAREABODY Properties 87
13 ROWTABAREA and COLTABAREA 89
ROWTABAREA Properties 91
COLTABAREA Properties 117
TABPAGE Properties 140
The Most Common Error 141
Example: Controlling which Tab is displayed by the Server Adapter 141
Example: Controlling the Visibility of Tab Pages 143
14 ROWTABLE0 and COLTABLE0 147
ROWTABLE0 Properties 149
COLTABLE0 Properties 151
15 ROWDYNAVIS and COLDYNAVIS 153
ROWDYNAVIS Properties 155
COLDYNAVIS Properties 157
Some Comments on Controlling the Visibility of Controls 159
16 ROWDIV and INNERDIV 161
When to Use ROWDIV and INNERDIV Containers 164
ROWDIV Properties 164
INNERDIV Properties 165
17 ROWSCROLLAREA 169
ROWSCROLLAREA Properties 171
18 HSPLIT and VSPLIT 175
Example for HSPLIT 176
Example for VSPLIT 177
HSPLIT Properties 178
VSPLIT Properties 180
SPLITCELL Properties 181
Defining the Split Size 182
19 HLINE and VLINE 183
VLINE Properties 185
HLINE Properties 186
20 Performance Optimization with Containers 187
21 ROWTABSUBPAGES and STRAIGHTTABPAGE 191
Adapter Interface 192
Built-in Events 193
Session Management 193

Performance Considerations 194
ROWTABSUBPAGES Properties 194
STRAIGHTTABPAGE Properties 196
III Working with Controls 197
22 Some Common Rules for all Controls 199
Name and Text ID 200
Table, Row, Column, Control 200
Explicit Alignment 200
Binding to Adapter Parameters 201
Directly Influencing the Control Style 201
Dynamically Controlling the Visibility and the Display Status of Controls 202
Focus Management 204
Flushing of Inputs 205
Tab Sequence 205
Tooltips 207
Images 208
Documents 209
23 BREADCRUMB 211
Example 212
Adapter Interface 212
Built-in Events 212
Properties 213
24 BUTTON 215
Example: Simple Button 216
Example: Button with Image 217
Hiding and Disabling Buttons 217
Properties 217
25 BUTTONLIST 225
Adapter Interface 226
Properties 226
26 CHECKBOX 229
Properties 230
27 COMBODYN2 235
Adapter Interface 236
Properties 236
28 COMBOFIX 243
COMBOFIX Properties 244
COMBOOPTION Properties 248
29 DATEINPUT 249
Example 250
Properties 250
30 DATEINPUT2 257
Customizing Date and Calendar Formats 258
DATEINPUT2 – Custom Dates 258

Properties 259
31 DROPICON 267
Example 268
Properties 268
32 FIELD 273
Built-in Events 274
Properties 274
33 FILEUPLOAD/FILEUPLOAD2 287
FILEUPLOAD 288
FILEUPLOAD2 289
FILEUPLOAD Properties 290
FILEUPLOAD2 Properties 294
34 ICON 297
Example 298
Properties 298
35 ICONLIST 305
Adapter Interface 306
Built-in Events 306
Properties 306
36 IHTML 309
Properties 310
37 IMAGEOUT 313
Properties 314
38 IMAGEVIEWER 317
Adapter Interface 318
Example 319
Properties 319
39 LABEL 323
Example 325
Aligning the Text 325
Properties 326
40 MENUBUTTON 331
Example 332
MENUBUTTON Properties 333
MENUITEM Properties 335
41 METHODLINK 337
Properties 338
42 MULTISELECT 343
Example 344
Adapter Interface 344
Properties 344
43 RADIOBUTTON 349
Properties 350
44 SCHEDULELINE 355
Properties 356

45 SLIDER 363
Example 364
Adapter Interface 365
Properties 365
46 STRIPSEL 371
Example 372
Properties 372
47 SUBCISPAGE2 377
Example 378
Adapter Interface 379
Session Management 379
Properties 380
48 SUBPAGE 383
Properties 384
49 TABSEL 387
Adapter Interface 388
Built-in Events 388
Properties 389
50 TABSTRIP2 391
Example 392
Adapter Interface 392
Built-in Events 392
Properties 393
51 TAGCLOUD 397
Example 398
Adapter Interface 399
Built-in Events 399
Properties 399
52 TEXT 403
Using the max* Properties 404
Properties 404
53 TEXTOUT 413
Example 414
Properties 414
54 TOGGLE 421
Properties 422
55 CHART 427
About the SVG and JPEG Formats 428
Example 428
CHART Properties 429
CHARTCOLUMN Properties 431
56 GOOGLEMAP 433
Before You Start 434
Example 434
Typical Problems 436

Properties 436
57 OPENSTREETMAP and Sub Controls 439
Example 440
Properties for OPENSTREETMAP 440
Properties for MAPMARKER 443
Properties for MAPLAYER 446
Properties for MAPPOINT 447
Properties for MAPLINE 449
Properties for MAPPOLYGON 450
58 REPORT 453
Example 454
Built-in Events 455
Properties 455
59 REPORT2 459
Example 460
Built-in Events 460
Properties 460
60 TIMER 461
61 NJX:BUTTONITEM 463
Example 464
Built-in Events 464
Properties 464
62 NJX:BUTTONITEMFIX 469
Example 470
Built-in Events 470
Properties 471
63 NJX:BUTTONITEMLIST 477
Example 479
Adapter Interface 479
Built-in Events 479
Properties 480
64 NJX:BUTTONITEMLISTFIX 481
Example 482
Adapter Interface 482
Built-in Events 483
Properties 483
65 NJX:DOCUMENTLINK 485
Properties 486
66 NJX:EVENTDATA 491
Example 493
Adapter Interface 494
67 NJX:FIELDITEM 495
Example 497
Adapter Interface 498
Built-in Events 498

Properties ... 498
68 NJX:FIELDLIST ... 507
Example ... 509
Adapter Interface ... 510
Built-in Events ... 510
Properties ... 510
69 NJX:FIELDVALUE ... 513
Example ... 515
Adapter Interface ... 515
Built-in Events ... 515
Properties ... 515
70 NJX:MASHZONE and BMOBILE:MASHZONE ... 525
Before You Start ... 526
Example ... 527
Adapter Interface ... 527
Properties NJX:MASHZONE ... 528
Properties BMOBILE:MASHZONE ... 530
Individual Data Sources per Session ... 530
71 NJX:NJXFILEDOWNLOAD ... 533
Example ... 535
Adapter Interface ... 535
Properties ... 535
72 NJX:NJXFILEUPLOAD2 ... 537
Example: File Upload to Natural ... 539
Adapter Interface: Upload to Natural Server ... 539
Adapter Interface: Upload to Filesystem ... 539
Built-in Events ... 540
Properties ... 540
73 NJX:NJXVARIABLE ... 543
Example ... 544
Properties ... 544
IV Working with Grids ... 545
74 Basics ... 547
75 TEXTGRID2 ... 549
A Simple Example ... 550
Adapter Interface for TEXTGRID2 ... 551
Selecting Rows in a TEXTGRID2 ... 551
TEXTGRID2 Properties ... 552
COLUMN Properties ... 558
Dynamic Setting of Text Styles in TEXTGRID2 ... 562
CSVCOLUMN Properties ... 564
76 TEXTGRIDSSS2 - TEXTGRID2 with Server-Side Scrolling ... 567
Performance Considerations ... 568
Example ... 568
Adapter Interface ... 570

Using Server-Side Scrolling 570
Using Server-Side Sorting 571
Setting the Client-Side Loading Behavior 571
TEXTGRIDSSS2 Properties 571
77 ROWTABLEAREA2 - The Flexible Control Grid 581
Example for ROWTABLEAREA2 582
Adapter Interface for ROWTABLEAREA2 584
Built-in Events for ROWTABLEAREA2 584
Making Grids Look like Grids 585
Making Columns Movable 586
Export to Clipboard and File 587
Icon Bars 589
ROWTABLEAREA2 Properties 596
STR Properties 603
78 ROWTABLEAREA3 - The Array Grid 607
Example for ROWTABLEAREA3 608
Adapter Interface for ROWTABLEAREA3 610
Built-in Events for ROWTABLEAREA3 613
ROWTABLEAREA3 Properties 613
TR3 Properties 619
GRIDCOLHEADER3 Properties 620
STR3 Properties 622
FIELD3 Properties 623
79 FLEXLINE - Flexible Columns in Control Grids 633
Example 634
Adapter Interface 636
ATTRIBUTES 636
FLEXLINE Properties 636
80 MGDGRID - Managing the Grid 639
Example 641
Adapter Interface 642
Built-in Events 643
MGDGRID Properties 643
ROWINSERT Properties 648
ROWCOPY Properties 648
ROWDELETE Properties 649
81 GRIDCOLHEADER - Flexible Column Headers 651
Flexible Column Sizing 652
Flexible Column Sorting 654
GRIDCOLHEADER Properties 655
Smart Selection of Rows - SELECTOR Control 659
SELECTOR Properties 660
82 Styling Grids 663
General Hints 664
Styling ROWTABLEAREA2, ROWTABLEAREA3 and Headers 664

V Working with Trees ... 669
83 Basics ... 671
Types of Trees ... 672
When to Use Which Type ... 673
84 TREENODE3 in Control Grid (ROWTABLEAREA2) ... 675
Example ... 676
Adapter Interface ... 677
Built-in Events ... 677
Properties ... 677
85 CLIENTTREE ... 683
Example ... 684
Adapter Interface ... 684
Built-in Events ... 685
Properties ... 685
VI Working with Menus ... 689
86 Types of Menus ... 691
87 MENU ... 693
Example ... 694
Adapter Interface ... 695
Built-in Events ... 695
Properties ... 695
88 DLMENU ... 699
Example ... 700
Adapter Interface ... 701
Built-in Events ... 701
Properties ... 701
89 XCIPOPUPMENU - Enable Context Menus ... 703
Example ... 705
Adapter Interface ... 706
Built-in Events ... 707
Properties ... 707
90 Styling Menus ... 709
Shared and Unshared Style Classes in Menu Controls ... 710
VII Non-Visual Controls and Hot Keys ... 713
91 AUTOCOMPLETE ... 715
General Information ... 716
Example ... 717
Data Sources for Populating the Values ... 717
Properties ... 720
92 TIMER ... 723
Example ... 724
Properties ... 725
93 XCIDATADEF - Data Definition ... 727
Example ... 728
Properties ... 731

94 XCICONTEXT 733
General Information 734
Example 734
Properties 735
95 NJX:XCIOPENPOPUP 737
Example 738
Adapter Interface 738
Non-responsive pages 740
Responsive pages 740
96 NJX:XCILIVINGPOPUP 741
Example 742
Adapter Interface 742
97 Extended Hot Key Management 743
Direct Hot Key Definitions with Certain Controls 744
Hot Key Definitions for Certain Controls 744
98 Function Key Handling 747
99 NJX:OBJECTS 749
General Information 750
Example 751
Adapter Interface 751
100 NJX:SESSIONPARAMS 753
General Information 754
Example 754
Adapter Interface 755
101 NJX:REQUESTCONTEXT 757
General Information 758
Adapter Interface 760
102 NJX:TRIGGEREVENT 761
Examples 762
Adapter Interface 764
VIII Working with Pop-Ups 765
103 Working with Pop-Ups 767
Pop-Ups - Non-Responsive Pages 768
Pop-Ups - Responsive Pages 772
IX Working with Workplaces 773
104 What are Multi Frame Pages? 775
105 Definition of Multi Frame Pages 777
MFPAGE 778
MFCISFRAME 779
MFHTMLFRAME 782
MFFRAMESET 783
106 Application Designer Workplace Framework 785
Framework Overview 786
Functions Frame: MFWPFUNCTIONS 788
Active Functions Frame: MFWPACTIVEFUNCTIONS 790

Content Frame: MFWPCONTENT 791
Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO 792
Customizing the MFWPFUNCTIONS Behavior 803
Session Management inside the Workplace 812
Workplace API for Dynamic Manipulation 812
107 Creating Your Own Workplace Application 815
Elements of a Workplace Application 816
Using the Default Workplace Framework 816
Customizing the Frames, Dialogs and Messages of the Default Workplace Framework 817
Using Your Own Standard Pop-up Dialogs and Messages 817
Using Your Own Active Functions Frame 819
108 Executing and Debugging Workplace Applications 821
Prerequisites 822
Creating a Session Configuration on the Application Server 822
Configuring the Session in Your Workplace Application 823
Executing the Workplace Application in NaturalONE 823
Debugging the Workplace Application in NaturalONE 823
Using Private-mode Libraries in Workplaces 825
109 Multi Language Management in Workplace Applications 827
General Information 828
Language Switch in Content Pages 829
Language Switch in Function Tree and Activities Pane 829
110 NJX:XCIWPINFO2 831
Example 832
Adapter Interface 832
111 NJX:XCIWPFUNCTIONS 837
Example 838
Adapter Interface 838
112 NJX:XCIWPACCESS2 845
Example 846
Adapter Interface 846
X Working with PDF Documents 849
113 Working with PDF Documents 851
General Information 852
About the Adapter Listener 852
Example 853
Built-in Events 854
Advanced Data Binding and Rendering 855
XI Working with Icons 859
114 Working with Icons 861
Using Own Icons 862
Using Bootstrap Icons 862
Using Bootstrap Icons as Icon Fonts 862

Using Bootstrap Icons as SVG Icon Files 864
Using SVG from Natural Programs 864

Natural Page Layout

Page Layout and Control Reference	
Typical Page Layout	Describes the elements used for the layout of a page.
Working with Containers	Shows you how to work with containers - containers are areas on the page that can hold controls.
Working with Controls	Shows you how to work with the elements that are placed into containers - the controls.
Working with Grids	Explains what grids are and how to use them.
Working with Trees	Explains the basic types of trees and how to use them.
Working with Menus	Shows you how to arrange a number of functions in a structured way.
Non-Visual Controls and Hot Keys	Describes how to develop controls that do not have visual effects.
Working with Pop-Ups	Describes how to develop pop-up controls.
Working with Workplaces	Deals with applications that organize multiple pages in so-called workplaces.
Working with PDF Documents	Explains how to create PDF documents for Natural page layouts.
Working with Icons	Explains how to use icons in controls and CSS

1 About this Documentation

- Document Conventions 2
- Online Information and Support 2
- Data Protection 3

Document Conventions

Convention	Description
Bold	Identifies elements on a screen.
`Monospace font`	Identifies service names and locations in the format *`folder.subfolder.service`*, APIs, Java classes, methods, properties.
`Italic`	Identifies: Variables for which you must supply values specific to your own situation or environment. New terms the first time they occur in the text. References to other documentation sources.
`Monospace font`	Identifies: Text you must type in. Messages displayed by the system. Program code.
{ }	Indicates a set of choices from which you must choose one. Type only the information inside the curly braces. Do not type the { } symbols.
|	Separates two mutually exclusive choices in a syntax line. Type one of these choices. Do not type the | symbol.
[ ]	Indicates one or more options. Type only the information inside the square brackets. Do not type the [ ] symbols.
...	Indicates that you can type multiple options of the same type. Type only the information. Do not type the ellipsis (...).

Online Information and Support

Product Documentation

You can find the product documentation on our documentation website at **https://documentation.softwareag.com**.

Product Training

You can find helpful product training material on our Learning Portal at **https://learn.softwareag.com**.

Tech Community

You can collaborate with Software GmbH experts on our Tech Community website at **https://techcommunity.softwareag.com**. From here you can, for example:

- Browse through our vast knowledge base.
- Ask questions and find answers in our discussion forums.
- Get the latest Software GmbH news and announcements.
- Explore our communities.
- Go to our public GitHub and Docker repositories at **https://github.com/softwareag** and **https://hub.docker.com/publishers/softwareag** and discover additional Software GmbH resources.

Product Support

Support for Software GmbH products is provided to licensed customers via our Empower Portal at **https://empower.softwareag.com**. Many services on this portal require that you have an account. If you do not yet have one, you can request it at **https://empower.softwareag.com/register**. Once you have an account, you can, for example:

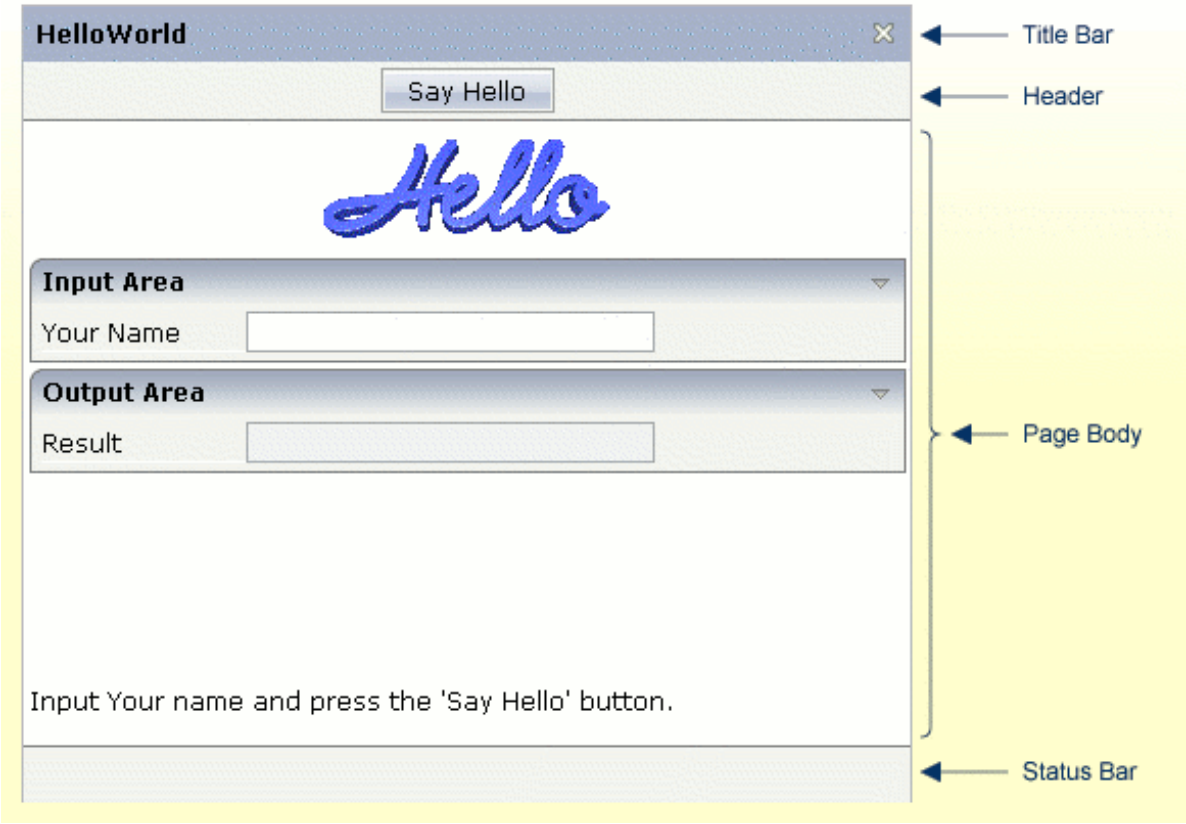
- Download products, updates and fixes.
- Search the Knowledge Center for technical information and tips.
- Subscribe to early warnings and critical alerts.
- Open and update support incidents.
- Add product feature requests.

Data Protection

Software AG products provide functionality with respect to processing of personal data according to the EU General Data Protection Regulation (GDPR). Where applicable, appropriate steps are documented in the respective administration documentation.

I Typical Page Layout

The layout of a page typically contains the following elements:



This part describes these elements in more detail.

NATPAGE
TITLEBAR
HEADER
PAGEBODY
STATUSBAR

2 NATPAGE

- Properties 8

The NATPAGE control is always the top node of a Natural page's layout definition. The Natural page, on the one hand, generates the visible container in which all the contained elements are placed; on the other hand, some Natural-specific settings are defined on page level.

Properties

Basic			
chartconfigfile	You can customize the rendering style of the charts in a separate JavaScript file. Specify where the file is located relative to your User Interface Component. Example: ./styles/MyOptions.js	Optional	
translationreference	This is the "translation reference" that is passed to the multi language management. The "translation reference" is a logical term representing a group of textids together with their translation. If using the standard file based multi language management that comes with CIS as default then a "translation reference" represents one file containing text-ids and translations in a comma separated format. Translation information is loaded by the multi language management "per translation reference". I.e. if a page links to a certain translation reference then all the translation information that is available through this reference is loaded in one step and is also buffered. You can set up different scenarios: either each page may address an own translation reference. E.g. if your page is named "abc.xml" then it references to "abc" - as consequence there is (per language) one abc.csv file holding translation information for this page. If you have a second page "def.xml" then you may define "def" accordingly. In this case each page is independent from the other. - On the other side you are required to translate certain "common text-ids" multiple times. If you on the other hand define one translation reference for multiple pages then you can share text-ids throughout the various pages. Please set up a strategy for using translation references when starting using the multi language management. The strategy should also include a structured way of naming text-ids. Text-ids may only be shared in an efficient way	Sometimes obligatory	

	if it is clear what they stand for. E.g. you may names of buttons in the following way: "btn_save" and "btn_saveas".		
stylesheetfile	URL of a style sheet file used for control rendering. Typically the style sheet file used for control rendering is set dynamically e.g. the style depends on the user who is currently logged on. When defining the style sheet file by this property, the style sheet file is not set dynamically but defined in a fix way for this page. The style sheet file must be defined as URL, relative to the generated page. A valid value may be "../softwareag/styles/CIS_DEFAULT.css". If not using the "hard setting" of the style sheet file via this property then the style sheet is determined by the runtime in the following way: (1) The adapter object provides for a "String getStyle()" method that return the URL. You can override the default method and pass back your own URL. (2) When using the default implementation derived from com.softwareag.cis.server.Model then the getStyle() method accesses the CIS session context. You can set the session's style by calling "findCISessionContext()" in your adapter and calling "setStyle()" in the session context's object.	Optional	css
responsivestylesheetfile	URL of the responsive stylesheet file for Bootstrap. If not specified the default Bootstrap stylesheet file is used. For more information see the responsive style guide. For non responsive pages this property is ignored.	Optional	
datatablesstylesheetfile	URL of the responsive DataTables stylesheet file for rendering responsive grids. If not specified the default DataTables stylesheet file is used. For more information see the responsive style guide. For non responsive pages this property is ignored.	Optional	
uselatestbootstrap	Set this to true if you want to use Bootstrap 4 for the responsive controls. It will overwrite the corresponding setting of cisconfig.xml for this page. Default is false.	Optional	
bootstrapversion	The bootstrapversion parameter in the cisconfig.xml defines the used bootstrapversion for the complete application. Only set this property if you want to overwrite the setting of the cisconfig.xml. Supported values are 4 and 5.	Optional	4 5
addstylesheetfile	URL of an additional style sheet file.	Optional	css

	You may use this additional style sheet file in order to define more styles than are provided in the "normal" style sheet file. Typical situations are: (A) Some controls offer the possibility to render defined content by style-class definitions (e.g. inside a TEXTGRID you can dynamically define which style-class is used for a certain cell). (B) If you define own controls by using the control extension framework and if these controls require own style classes then these style classes may be provided inside the additional style sheet file. By using the additional style sheet file you are able to avoid doing manipulations to the "normal" style sheet files that come from CIS or that are generated inside the tool "Style Sheet Editor".		
imagestopreload	Semicolon separated list of image-URLs that are directly preloaded in an invisible area of the page. If images are used inside a tree or a text grid then they are loaded by dynamically generated HTML that is placed into a corresponding area of the page. In order to optimise the loading you can preload such images by listing them in this property. The URL of the images must be relative to your generated HTML page. Example: if your page has a tree with certain node images then you may define: "images/nodeopened.gif" images/nodeclosed.gif; images/nodeendnode.gif".	Optional	
darkbackground	Normally a page background is in light colour (white if using CIS_DEFAULT style sheet). CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used if using the SUBCISPAGE tag or ROWTABSUBPAGES tag to seamlessly integrate inner pages into darker container areas.	Optional	true false
helpid	This is the id that is passed into the help management for the page. If a user clicks F1 inside the page and if there is no specific context sensitive control help available (e.g. help for field) then the help for the page is popped up.	Optional	
visiblevalueifundefined	Several CIS controls support a VISIBLEPROP property. The VISIBLEPROP contains the binding to an adapter	Optional	true false

	property that decides at runtime if a control is visible or not. This property defines how these controls behave if there is no implementation available for the property. Example: the VISIBLEPROP of a CHECKBOX is binding to a property "cbvisible" but there is not corresponding implementation "getCbvisible". If set to "true" then all controls with undefined visibility are displayed. If set to "false" then they are hidden.		
contextmenumethod	Name of the event that is sent to the adapter when the user clicks into the page with the right mouse button and no other control (e.g. textgrid, tree,...) handled the click so far.	Optional	
immediatedisplay	Flag that indicates if the screen is visible within the initial loading phase. Default is false. When using the default you see a light HTML page showing a "just loading" image. Use property "justloadingurl" to specify a page of choice.	Optional	true false
autotab	Sets the default behavior if an automatic tab should be executed for FIELD controls in this layout. Notice that this default is not used for FIELD controls in FLEXLINEs.	Optional	true false
itrinlinedisplay	Set this property if you want to set the property inline for all ITR controls of this page.	Optional	true false
buttonctrlenter	If set to TRUE, the method set in the button will be activated on ctrl enter even if a hotkey for ctrl enter has been set on page level. Default is FALSE.	Optional	true false
focusmgtprop	Name of adapter parameter that dynamically controls the focus management in the browser for the current server roundtrip. Valid values provided by the adapter are: 0 (=default), 1 (= suppress focus), 2 (= set focus), 3 (= open tabs in TABPAGE controls), 4 (= page refresh).	Optional	
addjavascriptlibs	Comma separated list of URLs of additional javascript libraries. Example: "../yourproject/js/yourlib.js". Used to include non-CIS javascript. Example of Usage: with the DATEINPUT control you can run own rules to convert and validate user input.	Optional	
flushmethod	Name of the event that is sent to the adapter in case the page loses the focus.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

adapterlisteners	Semicolon separated list of classes which connect to the server side adapter processing as adapter listeners (each one supporting the interface IAdapterListener).	Optional	
framebufferpriority	Priority (integer) that is used to manage the page within the CIS frame buffer. Use value "-1" to indicate that the page should not be buffered at all (typically used when having a FILEUPLOAD2 control on the page). Default is "0". Use any other integer value to indicate higher priority.	Optional	0 -1
centralcontextmenu	If set to 'true' then the context menu is rendered in a central frame. This central frame can be specified via the "popupdivframe" setting in cisconfig.	Optional	true false
withownborder	If set to "true" the page will be surrounded by an additional border.	Optional	true false
userinputprop	Name of the adapter parameter which will have a value of "true" if some userinput in the page or one of its subpages has been done since the last server-roundtrip.	Optional	
pdfpageprop	Name of the adapter parameter that dynamically defines the name of the pdf layout. This allows for applying different pdf layouts dynamically at runtime. If not specified a default name is used.	Optional	
Natural			
natsource	Specifies a name for the Natural adapter object that will later be generated from your page layout. During adapter generation, this name is checked to match the Natural naming conventions for objects. If you do not specify a name here, the adapter name is taken from the layout name. This might result in names that are not valid for Natural objects. These adapters can only be used in Natural for Eclipse.	Optional	
natdataarea	If this property is set, additional to the Natural adapter a separate Natural parameter data area (PDA) is generated for this layout. If this property is not set, no separate data area file is generated.	Optional	
natsinglebyte	Specifies whether string properties of the page are to be mapped to Unicode strings (U) or code page strings (A) in Natural. The value "true" means code page strings. The value "false" means Unicode strings (default).	Optional	true false
natkcheck	For many controls a data structure is generated into the Natural adapter. The user can choose the name of the data structure, but the variable names inside the data structure are predefined. If this property is set to “true”, it is guaranteed that these predefined names do not collide with Natural reserved keywords and conform to the checks that the Natural parameter KCHECK defines. If	Optional	true false

	this property is set to “false”, this is not guaranteed. For compatibility reasons the default value of the property is “false”.		
natrecursion	Properties of controls used in the page might have a recursive structure. These structures are mapped to multi-dimensional arrays in the Natural adapter. Natural arrays are limited to three dimensions. Therefore, the recursion depth of these structures can be limited using this property.	Optional	1 2 3 int-value
natdc	Specifies the character that is to be used as the decimal character in the format specifications of variables with decimal format in the parameter data area of the Natural adapter. For example, if a comma (,) is specified, "(N7,2)" is generated. If a period (.) is specified, "(N7.2)" is generated. The default is the period (.).	Optional	, .
natsss	The controls ROWTABLEAREA2 and MGDGRID support server-side scrolling and sorting. The corresponding data structures are generated into the parameter data area of the Natural adapter only if this attribute has been set to true. The default is false. This is for compatibility with earlier versions. For the control TEXTGRIDSSS2, the server-side scrolling data structures are always generated.	Optional	true false
natcv	Name of a Natural control variable that shall be assigned to the page. The control variable must be defined in a Data Definition (XCIDATADEF) control on the same page. The application can use the control variable to check the modification status of the page.	Optional	
xmlns:njx	Internal use only. Do not modify this.	Optional	
Popup			
popupwidth	Each page can be opened as a popup dialog. This property defines the pixel width preferred for the page. A popup is typically opened by the PROCESS PAGE MODAL statement in your Natural program. If no further definition is done then the popup will open in the width that is defined by this value. You can also dynamically manipulate the size and position of the popup by adding a NJX:XCIOPENPOPUP control to the parent layout.	Optional	100px 200px 300px 400px
popupheight	Each page can be opened as a popup dialog. This property defines the pixel height preferred for the page. A popup is typically opened by the PROCESS PAGE MODAL statement in your Natural program. If no further definition is done then the popup will open in the height that is defined by this value. You can also dynamically	Optional	100px 200px 300px 400px

	manipulate the size and position of the popup by adding a NJX:XCIOPENPOPUP control to the parent layout.		
popupfeatures	In addition to POPUPWIDTH and POPUPHEIGHT you can control the appearance of the popup dialog in which the current page may be displayed. You define a string to maintain different feature aspects, separated by semi-colon. For responsive pages closeonclick:true/false closeonesc:true/false withclosebutton:true/false sizetocontent:true/false resizable:true/false draggable:true/false For browser popups(deprecated) in non-responsive pages center:yes|no edge:sunken|raised resizable:yes|no scroll:yes|no status:yes|no (to display or hide a status bar) There is one special function built in by which you can position a popup relative to its caller's window (the dialogLeft and dialogTop definition normally refer to absolute coordinates of the screen): by specifying "dialogLeft: SCRX(100)px" you define that the position is 100 pixels right from the left top corner of the current window. - Use "dialogTop: SCRY(100)px" in the same way for vertical positioning.	Optional	dialogLeft: 200px dialogTop: 100px edge: sunken resizable: yes status: no
popupendmethod	When set to TRUE the event nat:popup.end is raised when the page is running as popup. Default value is FALSE.	Optional	true false
Occupied			
occupiedimage	URL of the image that is displayed to indicate that the screen is just communicating to the server. This is the image that is located in the top left corner and which by default is a flashing hour glass.	Optional	

	You can specify any image, e.g. also animated GIF files. If you want your image not to be visible in the top left corner but "somewhere" in the screen then draw an image with some transparent area on the left and above the image that you want to show.		
occupiedpixelheight	When the screen is busy, because the client is exchanging information with the server, an hour glass image is displayed at the top left corner. With this property you define the pixel height of this hour glass image.	Optional	
occupiedpixelwidth	When the screen is busy, because the client is exchanging information with the server, an hour glass image is displayed at the top left corner. With this property you define the pixel width of this hour glass image.	Optional	
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	
delayhotkeys	Set this property to true, if the hotkey should be executed delayed.	Optional	true false
Loading			
justloadingurl	URL of the page that is displayed to indicate that screen is just loading. Typically this is a light HTML page showing a loading image of choice. Use plain HTML - not a generated CIS page.	Optional	

3 TITLEBAR

- Properties 18

The title bar is typically placed at the top of a page. The text in the title bar can either be set statically inside the layout definition, or it can be dynamically resolved by a property of the corresponding adapter.

The title bar can have a close icon (cross at the top right) and an online help icon.The close icon triggers the `nat:page.end` event in the Natural application.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
withclose	In the right top corner of the titlebar there is by default a close-icon. Define "false" in this property in order to hide this icon. The close-icon calls the method "endProcess" of your adapter. "endProcess" is implemented in the class "com.softwareag.cis.server.Model" and by default ends the subsession the adapter is running in. - Override this implementation if this default implementation does not fit to your needs.	Optional	true false
align	Horizontal alignment of the text that is shown.	Optional	left center right
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying	Optional	

	"../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
helpid	Id that is passed to the online help management. If this "helpid" is specified then a help-icon will be displayed in the right top corner. If clicking on the icon then the corresponding help will show up.	Optional	
titlestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
pixelheight	Height of the control in pixels.	Optional	1 2 3 int-value
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
closetitle	The text that is entered here appears as tooltip on the close-icon on the right top border of the titlebar.	Optional	
closetitletextid	Multi language dependent text that displays the tooltip on the close-icon. Do not specify a CLOSETITLE if you are specifying a CLOSETITLEID.	Optional	

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	Name of the adapter parameter that provides a value from which the titlebar text is dynamically derived. Do not use "name" or "textid" when using this "valueprop" property.	Optional	
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
imageprop	Name of adapter parameter which dynamically provides the URL of the image that is shown inside the control. The URL must either be an absolute URL or a relative URL.	Optional	
withcloseprop	Name of the adapter parameter that indicates if the close icon of the titlebar is visible. The server side property will be of type (L).	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	

njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

4 HEADER

- Properties 24

The header is an area in which you can place buttons, icons and menus. The area itself is grey and has a dark grey line at its bottom (if using the standard style sheet). The header is used to display buttons and icons that are valid for the whole page. Typically, it is placed directly under the title bar.

Properties

Basic			
nocellspacing	Flag that indicates if there is space between controls within the the header table. Default is FALSE.	Optional	true false
align	Horizontal alignment of the control's content. Default is "center".	Optional	left center right
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
withdistance	If set to TRUE then an additional distance will be added at the bottom of the header. Default is FALSE.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

5 PAGEBODY

- Properties 26

The page body is the main area in which you place the body part of your layout. The body adapts its height to the current window's height, while elements such as TITLEBAR, HEADER and STATUSBAR keep a constant height. If the page body's size is too small to hold its content, you scroll through the elements that are inside the PAGEBODY.

Properties

Basic			
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.	Optional	true false
pagebodystyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Padding			
horizdist	Defines if there is always a small horizontal distance kept between the border of the PAGEBODY area and its content. Set to 'false' if you want controls in the page body to directly start at the very left and to end at the very end - without any distance. Default is 'true'.	Optional	true false
paddingleft	Number of pixels which you want to keep as margin between the pagebody's border and its content. If you want that all contents inside your page body keeps a horizontal distance of 50 pixels on the left then specify: PADDINGLEFT = 50 The PADDINGLEFT and PADDINGRIGHT values are added in addition to the small horizontal distance which is added via the HORIZDIST property.	Optional	1 2 3 int-value
paddingright	Number of pixels which you want to keep as margin between the pagebody's border and its content. If you want that all contents inside your page body keeps a horizontal distance of 50 pixels on the right then specify: PADDINGRIGHT = 50 The PADDINGLEFT and PADDINGRIGHT values are added in addition to the small horizontal distance which is added via the HORIZDIST property.	Optional	1 2 3 int-value
paddingtop	Number of pixels which you want to keep as margin between the pagebody's border and its content. If you want that all contents inside your page body keeps a vertical distance of 50 pixels on the top then specify: PADDINGTOP = 50	Optional	1 2 3 int-value
paddingbottom	Number of pixels which you want to keep as margin between the pagebody's border and its content. If you want that all contents inside your page body keeps a vertical distance of 50 pixels on the bottom then specify:	Optional	1 2

	PADDINGBOTTOM = 50		3 int-value
Logon Form			
withformtag	Default value is false. If set to true all controls included in the pagebody tag will be surrounded by a form tag - only in the generated html page. That makes it possible to save or transfer forms. i.e. save username and password or a complete search form. You will also need an 'submitbutton' - please have a look at the button control.	Optional	true false

6 STATUSBAR

- General Information 30
- Example 30
- Properties 31

General Information

Normally, the status bar is located at the bottom of a page. It is a grey area (if using the standard style sheet) where status information can be seen. The status information is derived dynamically from the parameters sent with the `nat:page.message` event (see *Sending Events to the User Interface*). The information consists of three parts:

- Type of the status message - whether it is an error message (E), a warning (W) or a success message (S). Depending on the type, a small icon is displayed to the left of the message.
- The status message itself - the text displayed within the status message.
- A long text for the status - optional text shown in a dialog when clicking on the status message.

As an alternative to applying the status information as parameters of the `nat:page.message` event, you can apply your own `typeprop`, `shorttextprop` and `longtextprop` properties to the STATUSBAR control. This will generate the corresponding fields in your Natural variable. At runtime, you can apply the corresponding values in the usual way. Applying values to the generated fields has the same effect as sending the parameter values with the `nat:page.message` event. You can even mix both methods.

Example

In the "Hello World!" application of the Natural for Ajax demos (`HELLOW-P.NSP`), you want to display an error message if the user clicks the **Say Hello!** button and has not yet entered a name.

```
DECIDE ON FIRST *PAGE-EVENT
...
  VALUE U'onHelloWorld'
  IF YOURNAME = ' '
    PROCESS PAGE UPDATE FULL AND SEND EVENT 'nat:page.message' WITH
      PARAMETERS
        NAME 'type'  VALUE 'E'
        NAME 'short' VALUE 'Please enter your name'
      END-PARAMETERS
  ELSE
      COMPRESS "HELLO WORLD" YOURNAME INTO RESULT
      PROCESS PAGE UPDATE FULL
  END-IF
...
```

The screen including the error message looks as follows:



Properties

Basic			
typeprop	Name of the adapter parameter that provides as value the type of the status message. The type defines the image that is rendered at the beginning of the message. Currently there are 3 supported values: E for error, W for warning, S for success. Please pay attention: Do not use the name messageType. This name is internally used when no property name is specified.	Optional	
shorttextprop	Name of the adapter parameter that provides as value the message text that is visible inside the status bar. Please pay attention: Do not use the name messageShortText. This name is internally used when no property name is specified.	Optional	
longtextprop	Name of the adapter parameter that provides as value the long message text. The long text pops up if clicking onto the short text message. Please pay attention: Do not use the name messageLongText. This name is internally used when no property name is specified.	Optional	
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation.	Optional	true false

	Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".		
resetbefore	If set to TRUE, the control is reset before a server roundtrip is done.	Optional	true false
withdistance	If set to TRUE then an additional distance will be added at the top of the statusbar. Default is FALSE:	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

II Working with Containers

Containers are areas on your screen that can hold controls (such as fields, labels, etc.) or other container(s). Containers are the preferred way to structure elements inside your page body.

The information provided in this part is organized under the following headings:

Positioning of Controls inside a Container
Defining the Width of Controls inside a Container
Vertical Sizing of Containers and Controls
Overview of Different Containers
ROWAREA and COLAREA
ROWAREAWITHHEADER
ROWTABAREA and COLTABAREA
ROWTABLE0 and COLTABLE0
COLDYNAVIS and ROWDYNAVIS
ROWDIV and INNERDIV
ROWSCROLLAREA
HSPLIT and VSPLIT
HLINE and VLINE
Performance Optimization with Containers
ROWTABSUBPAGES and STRAIGHTTABPAGE

7 Positioning of Controls inside a Container

- Row Types - TR and ITR 36
- Some More Details on ITR 37
- ITR in Google Chrome and Edge Chromium 38
- TR Properties 39
- ITR Properties 40

Containers internally build an HTML table in which you place rows. Inside each row you place the controls - or again container(s).

Row Types - TR and ITR

There are two types of rows:

- The TR row is a normal table row. If you place more table rows - one under the other - inside one container, the columns inside the table row are all synchronized. See the example below in order to understand what “synchronized” means.

  Since controls are placed into columns, all controls are positioned in a synchronized way.
- The ITR row is a special table row. If you place more ITR table rows - one under the other - inside one container, each row has an independent set of columns; i.e. columns are not synchronized.

Have a look at the following XML layout description:

```
<rowarea name="With TR">
    <tr>
        <label name="First Name" width="100">
        </label>
        <field valueprop="fname" width="200">
        </field>
    </tr>
    <tr>
        <label name="Last Name" width="200">
        </label>
        <field valueprop="lname" width="200">
        </field>
    </tr>
</rowarea>
<rowarea name="With ITR">
    <itr takefullwidth="true">
        <label name="First Name" width="100px">
        </label>
        <field valueprop="fname" width="200">
        </field>
    </itr>
    <itr takefullwidth="true">
        <label name="Last Name" width="200">
        </label>
        <field valueprop="lname" width="200" length="20">
        </field>
    </itr>
</rowarea>
```

Note that each control (label, button, fields, etc.) is placed into one column of its own. If you have many controls inside one row - and have several rows one below the other - synchronized columns (using TR rows) sometimes cause funny results.

What is better, TR or ITR? Of course, it depends. The recommendation is:

- Use ITR as default. Using ITR, each row is defined independently from other rows that are positioned in the same container. You can change the number of controls (i.e. you internally change the number of managed columns) in one row without interdependencies to other rows.
- Only use TR if you really want to synchronize columns. A typical area of usage is inside the grid management (ROWTABLEAREA2 control): in a grid you explicitly desire to have synchronized columns inside the grid's table.

Some More Details on ITR

There are two ROWAREA containers. The first one uses TR rows, the second one uses ITR rows. The label for **First Name** has a width of 100 pixels, the label for **Last Name** has a width of 200 pixels. Now look at the result:



Inside the TR rows, all columns are synchronized - while in the ITR rows, each row is individually arranged.

How does the ITR control work internally? For each row, an individual table is opened with one row. Example: you define the following area in the XML layout definition:

```
<area>
  <itr>
    ...
    ...
  </itr>
  <itr>
    ...
    ...
  </itr>
</area>
```

The generated HTML looks like this:

```
<table>
  <tr>
    <td colspan="100">
      <table>
        <tr>
          ...
          ...
        </tr>
      </table>
    </td>
  </tr>
  <tr>
    <td colspan="100">
      <table>
        <tr>
          ...
          ...
        </tr>
      </table>
    </td>
  </tr>
</table>
```

Inside each row there is a table definition of its own, holding exactly one row.

You can define a `takefullwidth` property with the ITR definition, defining the width of the internal table of an ITR tag. If the `takefullwidth` property is set to "true", this means that the internal table that is kept per row is internally opened to use 100% of the available width. Without any definition, the table will be as big as it is required by its content.

ITR in Google Chrome and Edge Chromium

When using pixel sizing for controls in ITRs you may see rounding issues when zooming the page in Google Chrome and Edge Chromium. See the example "Inline rendering" in the NaturalAjax-Demos.

In case your ITR only contains the following controls: FIELD, LABEL, HDIST, ICON, BUTTON and/or XCIDATADEF, you can set the inline property to true. This will force the browser to use a different rendering style, which avoids the rounding issues while zooming. Instead of setting the inline property directly in the ITR you can specify this inline rendering for the whole page or the whole application.

To render this kind of ITRs of the whole page inline, set

```
<natpage itrinlinedisplay='true' ...
```

To render this kind of ITRs in the whole application inline, in *cisconfig.xml*, set

```
<cisconfig itrinlinedisplay='true' ... ↩
```

TR Properties

Basic			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). Please note: the row content may overrule this setting. The height setting "100px" of an embedded textbox will beat a row height of "50px". (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
withalterbackground	Flag that indicates if the grid line shows alternating background color (like rows within a textgrids). Default is false. Please note: controls inside the row must have transparent background. In case of the FIELD control simply set property TRANSPARENTBACKGROUND to true.	Optional	true false
trstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
trstyleprop	Name of the adapter parameter that dynamically provides explicit style information for the control.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

ITR Properties

Basic			
takefullwidth	If set to "true" then the control takes all available horizontal width as its width. If set to "false" then the control does not have a predefined width but grows with its content.	Optional	true false
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
align	Alignment of the content of the ITR row.	Optional	left center

	Background: the ITR as independent table row renders a table into its content area. Inside this table a row is opened in which the controls are placed. This table normally is starting on the left of the ITR row. With this ALIGN property you can explicitly define the alignment of the table.		right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.	Optional	true false
inline	Only set this property to true if you see rounding issues when zooming your page in Google Chrome or Edge Chromium browser. The property will force the browser to use a different rendering style for this itr. Use this property only if your ITR only contains the following controls: FIELD, LABEL, HDIST, ICON, BUTTON and/or XCIDATADEF and you are using pixel sizing.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Visibility			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Appearance			

itrstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
itrclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
itrstyleprop	Name of the adapter parameter that dynamically provides the style of the control.	Optional	

8 Defining the Width of Controls inside a Container

- Controlling the Width of Controls 44
- HDIST and VDIST Controls 46
- HDIST Properties 48
- VDIST Properties 48
- rowspan and colspan Definitions 49
- CELLSPAN Control 50
- CELLSPAN Properties 51
- Rules for Positioning Controls inside Containers 53

As mentioned in the previous section, each control is automatically embedded into a column. Consequently, the width of the control is, on the one hand, determined by the size of the control itself - on the other hand, the column is part of a table row and also follows the table row's sizing.

Controlling the Width of Controls

Every control that allows width sizing offers a corresponding `width` property. In this property, you put either an absolute pixel value (`width="100"`) or a percentage value (`width="50%"`). The rendering follows the strategy:

- If the width of a control is specified as a pixel value, the width is fixed: if the browser screen is too small to display all controls, the controls will not be reduced but keep their pixel size. Depending on your settings in the PAGEBODY tag (`hscroll` property), the displayed elements will be cut off or will be accessible by a scroll bar.
- If the width of a control is defined as a percentage value (`width="50%"`), HTML renders the control accordingly. If the screen is too small to show all controls, the browser will try to reduce elements according to the table rendering rules.

If you define the width of a control as a percentage value, the width relates to

- the width of the area in case of using TR rows, or to
- the width definition of the ITR row if using ITR rows. This width definition can either be absolute or percentage-based.

The following example shows a page in which controls hold percentages values for the width:

```
<itr takefullwidth="true">
    <label name="Factor1" width="20%">
    </label>
    <field valueprop="factor1" width="80%">
    </field>
</itr>
<itr takefullwidth="true">
    <label name="Factor2" width="20%">
    </label>
    <field valueprop="factor2" width="60%">
    </field>
    <hdist width="20%">
    </hdist>
```

The HTML page looks as follows - the size of the controls changes according to their percentage definition:



A similar screen is now built using absolutely defined pixel sizes:

```
<itr takefullwidth="false">
    <label name="Factor1" width="100">
    </label>
    <field valueprop="factor1" width="200">
    </field>
</itr>
<itr takefullwidth="true">
    <label name="Factor2" width="100">
    </label>
    <field valueprop="factor2" width="150">
    </field>
</itr>
```

In the ITR definition, there is no `width` specification - therefore, the controls will occupy exactly the space they require. The result looks as follows - the size of the controls will not change when changing the screen size:



Pay attention to what was said previously: Controls are placed into columns; columns are placed into table rows; and table rows are placed into containers. If you place a control into a row and define this control to have a width of 100%, then the elements “above” have to take care of providing the space to which the control relates its "100%". More concrete: If you place a FIELD control with a width of 100% into an ITR row that does not provide for a width of 100% itself (using the property `takefullwidth`), then the result will be a minimum-width field (100% of nothing).

Pixel sizing represents a bottom-up sizing approach: a control defines its width - all the other controls around (e.g. the container in which the control is placed) have as a consequence to adapt to the control's size: if the control is defined to occupy more space, then the container has to follow and provide for the space.

Percentage sizing represents a top-down sizing approach: the inner control tells how many percentages of the space that is granted from the outer control is occupied. As a consequence the outer control needs to define its size properly. Either the outer control itself defines a pixel size or it itself defines a percentage size - thus passing the responsibility to the next higher level. This might end up in a cascading definition of “percentage sizing” - up to the PAGEBODY control, which is the outer-most container of a page.

There are four commonly used properties for sizing:

- `width/height` - this is the quite obvious definition as explained in this section.
- `takefullwidth/takefullheight` - this is an equivalent to `width="100%"` and `height="100%"`.

HDIST and VDIST Controls

HDIST means “horizontal distance”. VDIST means “vertical distance”.

HDIST Control

The HDIST control represents a distance to be placed between controls. The distance itself holds a certain width that again can either be a pixel width or a percentage width.

The following example shows a table row into which a town and a zip code is put:



Between the two FIELD controls, you see a small distance that separates the fields from one another. The corresponding XML layout definition is:

```
<rowarea name="HDIST Example">
    <itr>
        <label name="Zip Code / Town" width="120">
        </label>
        <field valueprop="zipcode" width="80">
        </field>
        <hdist width="5">
        </hdist>
        <field valueprop="town" width="200">
        </field>
    </itr>
</rowarea>
```

The HDIST control is also very useful for percentage-based sizing of widths. If you want a control to occupy 50% of the available width, you have to “fill the gap” in the following way:



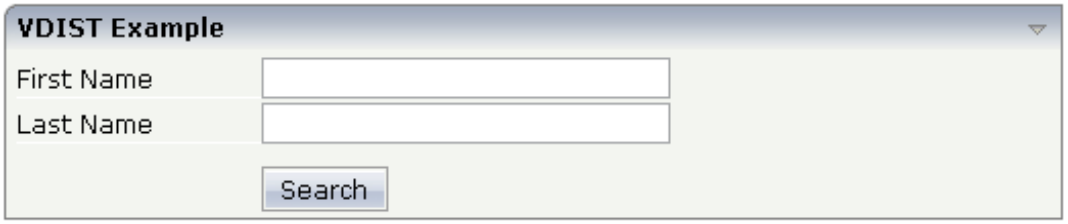
The corresponding XML layout definition is:

```
<rowarea name="HDIST Example">
    <itr height="100%">
        <label name="First Name" width="120">
        </label>
        <field valueprop="fname" width="50%">
        </field>
        <hdist width="50%">
        </hdist>
    </itr>
</rowarea>
```

Pay attention: when using percentage sizing, then you should take care of filling the "100%" by the controls inside the row. Otherwise, the browser will distribute the remaining space to its columns - i.e. the controls will not be positioned the way you expect.

VDIST Control

The VDIST control is the counterpart of the HDIST control - in vertical direction. The following example shows a scenario in which the line containing the BUTTON control keeps a vertical distance of 10 pixels from the lines containing the FIELD controls:



The layout definition is:

```
<rowarea name="VDIST Example">
    <itr height="100%">
        <label name="First Name" width="120">
        </label>
        <field valueprop="fname" width="200">
        </field>
    </itr>
    <itr height="100%">
        <label name="Last Name" width="120">
        </label>
        <field valueprop="lname" width="200">
        </field>
    </itr>
    <vdist height="10">
    </vdist>
    <itr>
        <hdist width="120">
        </hdist>
```

```
        <button name="Search" method="onSearch">
        </button>
    </itr>
</rowarea>
```

Note that an HDIST control is used in the line containing the BUTTON control to align the button to the fields.

HDIST Properties

Basic			
width	Width of the HDIST control, either in pixels or as percentage value. If no width is defined then a default width of 2 pixels is assigned.	Optional	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	

VDIST Properties

Basic			
height	Height of the VDIST control, either in pixels or as percentage value. If no width is defined then a default width of 3 pixels is assigned.	Optional	100 150 200 250

			300 250 400 50% 100%
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

rowspan and colspan Definitions

Each control has a `colspan` and `rowspan` property that is "1" by default. This definition is directly transferred to the column definition that is placed around the control.

Example:

```
<tr>
  <control colspan="2">
  </control>
</tr>
```

If you specify the above definition, the created HTML code looks like this:

```
<tr>
  <td colspan="2" rowspan="1">
    ... control-specific HTML code ...
  </td>
</tr>
```

The usage of `rowspan` and `colspan` only makes sense in scenarios in which you define multiple rows inside one container and if you use TR rows at the same time. You do not have to pay attention to them if working in ITR rows.

Again: first check if the TR way of arranging controls is really the best approach - compared to the ITR approach. Using TR means you have to “fight” with `colspan` and `rowspan` definitions in order to properly lay out your controls. With ITR, each row is independently defined from its neighbor rows.

CELLSPAN Control

Inside one row, you can place controls or nested containers. Containers again allow you to specify new rows inside the container.

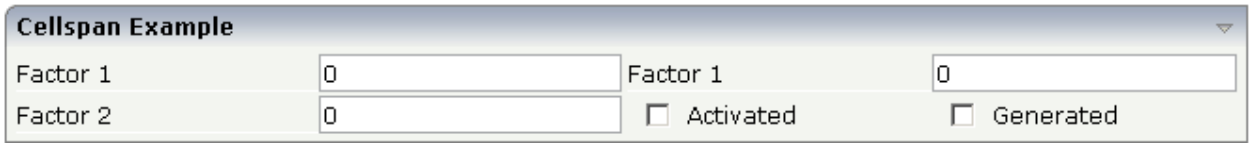
There is a special control, the CELLSPAN control. With the CELLSPAN control, you can quickly define one cell inside a row of a container to place other controls. The CELLSPAN control has a `width` property to specify the width of its inner content.

Have a look at the following example:

```
<rowarea name="Cellspan Example">
    <tr>
        <label name="Factor 1" width="25%">
        </label>
        <field valueprop="factor1" width="25%">
        </field>
        <hdist></hdist>
        <cellspan width="50%">
            <label name="Factor 1" width="50%">
            </label>
            <field valueprop="factor1" width="50%">
            </field>
        </cellspan>
    </tr>
    <tr>
        <label name="Factor 2" width="25%">
        </label>
        <field valueprop="factor2" width="25%">
        </field>
        <hdist></hdist>
        <cellspan width="50%">
```

```
            <checkbox valueprop="activated" width="10%">
            </checkbox>
            <label name="Activated" width="40%" asplaintext="true">
            </label>
            <checkbox valueprop="generated" width="10%">
            </checkbox>
            <label name="Generated" width="40%" asplaintext="true">
           </label>
        </cellspan>
    </tr>
</rowarea>
```

Each TR row contains one CELLSPAN definition with a width of 50%. The inner content of the CELLSPAN definitions is completely different between the rows:



You could add controls to the CELLSPAN definition in the first row without any implications inside the second row. The CELLSPAN control internally operates similar to the ITR control: it builds a table on its own and decouples its content from the surrounding table rendering.

CELLSPAN Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%

height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspanprop	Name of the adapter parameter which dynamically provides a colspan value at runtime.	Optional	
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows).	Optional	1 2 3 4 5

	It does not make sense in ITR rows, because these rows are explicitly not synched.		50 int-value
cellstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
backgroundclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	

Rules for Positioning Controls inside Containers

This is a collection of rules you should consider when positioning controls inside containers:

- Make up your mind where to use relative percentage values or absolute pixel definitions.
- Do not mix percentage and pixel values inside one container.
- Internally, Application Designer controls are mapped to the HTML tags `TABLE`, `TR` and `TD`. When developing, you should have in mind the normal HTML table management.
- Structure your container not as one big container holding one complex table, each row holding a lot of controls. Instead, use the possibility to define nested containers or CELLSPAN controls in order to structure your layout.

9 Vertical Sizing of Containers and Controls

- Vertical Pixel Sizing 56
- Vertical Percentage Sizing 57
- Finishing the Example 59

Nearly all controls which can be sized offer vertical sizing by a corresponding `height` property. You can set the value of this property either as a pixel value or as a percentage value.

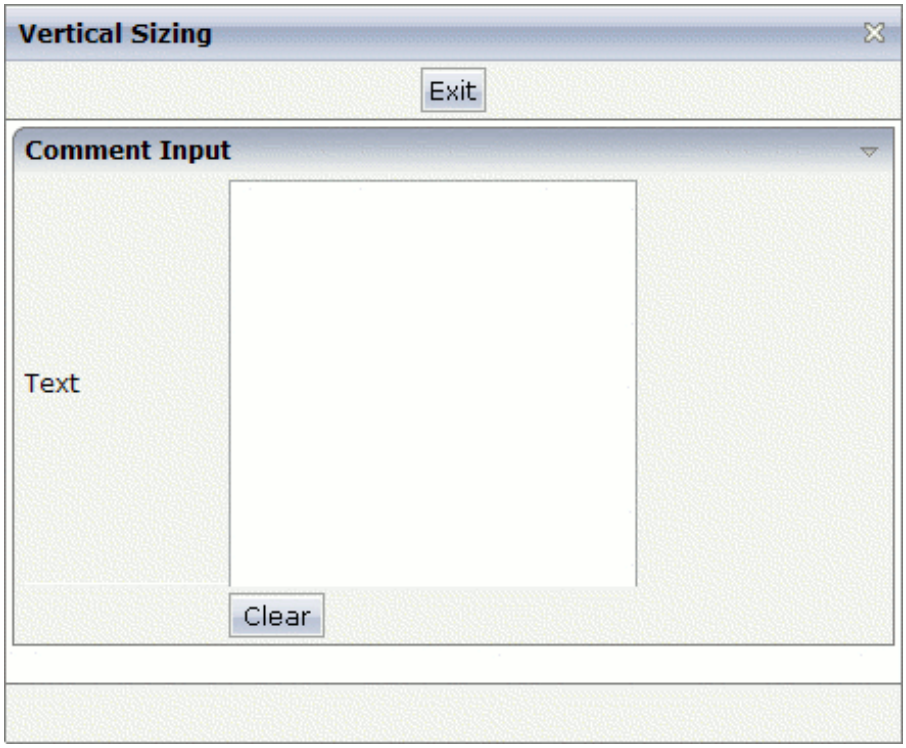
Vertical Pixel Sizing

This is the default. Controls either occupy their standard height or the height is explicitly defined in pixels. The whole page is sized from the bottom to the top.

Look at the following example:

```
<pagebody>
    <rowarea name="Comment Input">
        <itr>
            <label name="Text" width="100">
            </label>
            <text valueprop="comment" width="200" height="200">
            </text>
        </itr>
        <vdist>
        </vdist>
        <itr>
            <hdist width="100">
            </hdist>
            <button name="Clear" method="onClear">
            </button>
        </itr>
    </rowarea>
</pagebody>
```

The corresponding screen looks as follows:



The vertical size of the ROWAREA is exactly as big as required by its content. The TEXT control is defined to be 200 pixels high.

Vertical Percentage Sizing

Use the same example, but this time the size of the TEXT control should be as big as possible - depending on the size of the browser window. It should take the full available height.

The XML layout definition looks as follows:

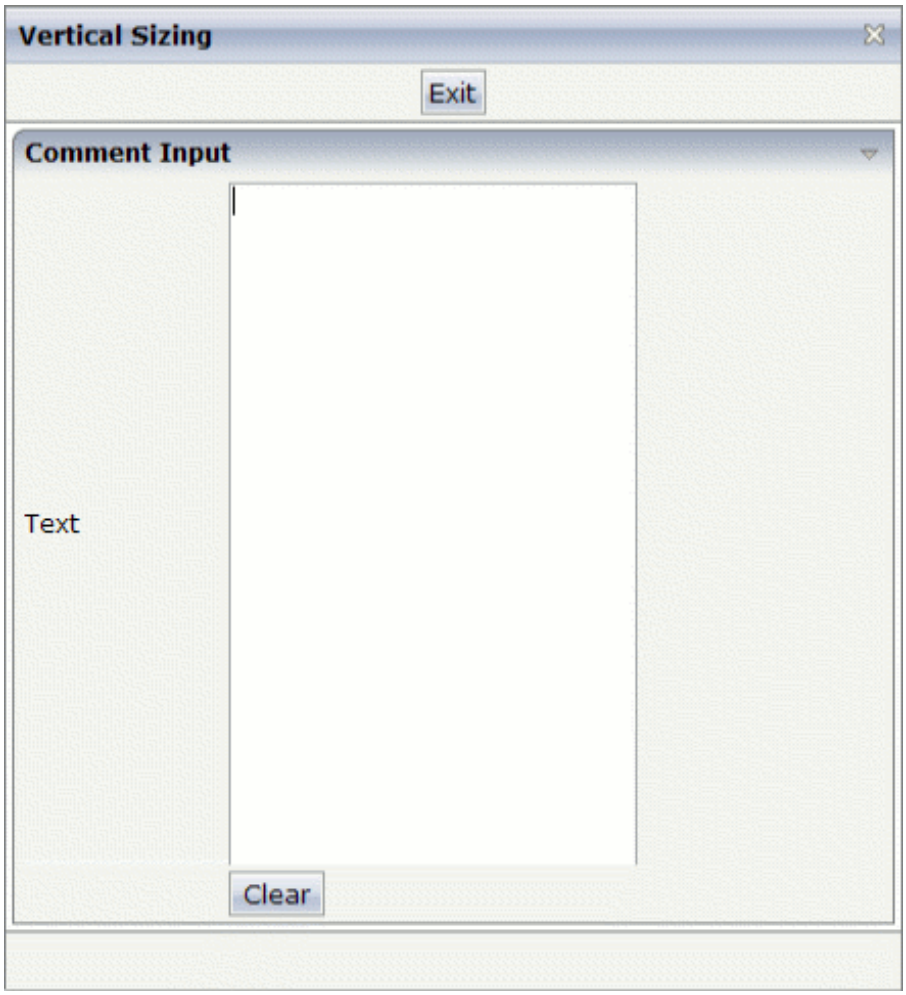
```
<pagebody takefullheight="true">
    <rowarea name="Comment Input" height="100%">
        <itr height="100%">
            <label name="Text" width="100">
            </label>
            <text valueprop="comment" width="200" height="100%">
            </text>
        </itr>
        <vdist>
        </vdist>
        <itr>
            <hdist width="100">
            </hdist>
            <button name="Clear" method="onClear">
```

```
            </button>
        </itr>
    </rowarea>
    <vdist>
    </vdist>
</pagebody>
```

The TEXT control now occupies a height of 100%. However, the definition of the whole size of the page is passed down from the PAGEBODY to the control:

- In the PAGEBODY, the property `takefullheight` is set to "true". This means that the content of the page body gets passed 100% of the available height.
- On the next level, the ITR row - in which the TEXT control is placed - is defined to have a height of "100%". This means it tries to grab as much height as possible. On the same level, there is also a VDIST (vertical distance) control and another ITR row - with no height defined. This means that these controls get as much height as they require due to their content - but the whole remaining vertical space is assigned to the first ITR row with the HEIGHT of "100%".

The result page looks as follows:



By changing the size of the browser window, the height of the whole control arrangement will follow accordingly.

You see that sizing by percentage values means that you have to think from top to bottom - just the opposite direction as you think with pixel values. This is nothing new for you if you are used to work with normal HTML tables - in fact, everything that is done below the diverse container controls is done by table rendering.

Conclusion: The example shows you that the `height` property of controls can be defined as a percentage value - but needs an outside reference to depend on. Some of the controls, such as the PAGEBODY, do not offer explicitly a `height` property but only a property `takefullheight` that can be set to "true". This is equivalent to a definition of `HEIGHT="100%"`.

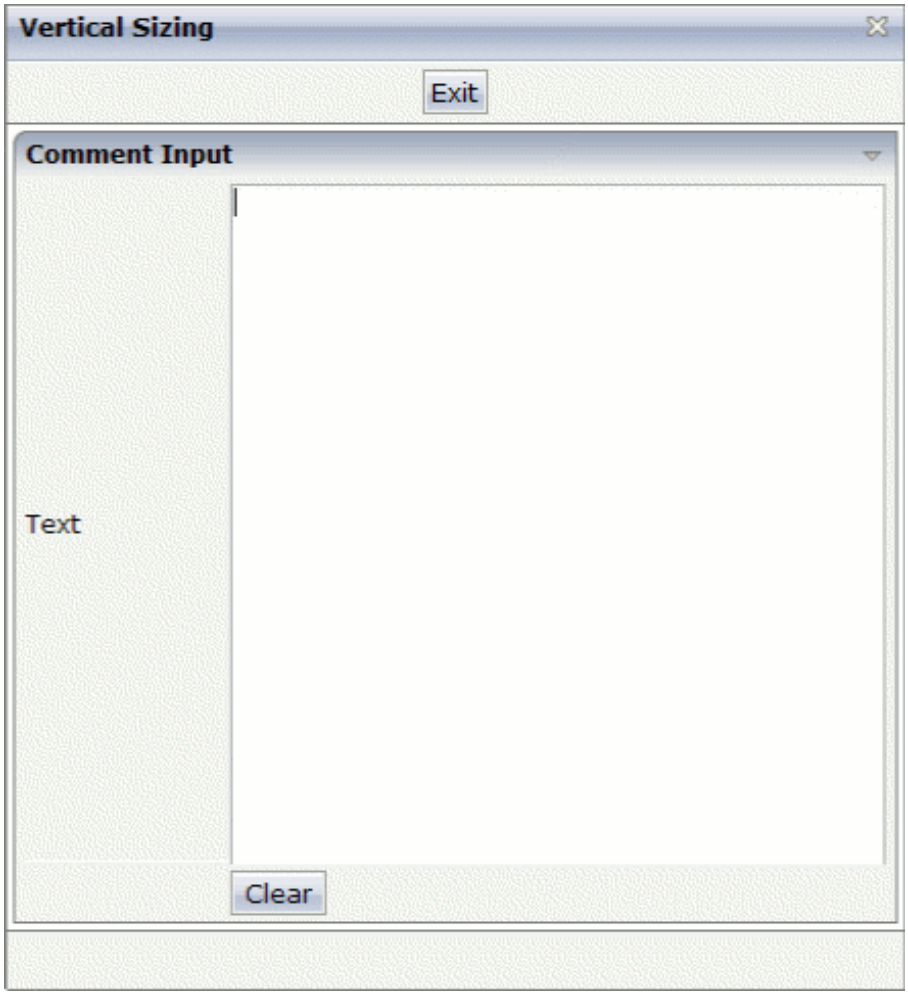
Finishing the Example

This has nothing to do with vertical sizing, but with horizontal sizing. We cannot finish the example without having changed it also in a way that it occupies the full available horizontal width. The layout definition now looks as follows:

```
<pagebody takefullheight="true">
    <rowarea name="Comment Input" height="100%">
        <itr takefullwidth="true" height="100%">
            <label name="Text" width="100">
            </label>
            <text valueprop="comment" width="100%" height="100%">
            </text>
        </itr>
        <vdist>
        </vdist>
        <itr>
            <hdist width="100">
            </hdist>
            <button name="Clear" method="onClear">
            </button>
        </itr>
    </rowarea>
    <vdist>
    </vdist>
</pagebody>
```

The `width` property of the TEXT control is set to "100%". Similar to the vertical height management, the available width is passed from the ITR row definition above - which occupies 100% of the available width inside the ROWAREA. The ROWAREA always occupies the whole available width - it does not require an explicit width definition.

The result is now:


Vertical Sizing
Exit
Comment Input
Text
Clear

10 Overview of Different Containers

- Different Kind of Containers 62
- Row Containers 62
- Column Containers 63
- Row and Column Containers in Combination 64
- Nesting Containers 65

Different Kind of Containers

Currently, there are the following types of containers:

- **ROWAREA and COLAREA**
  These are containers holding a title. The graphic area represented by the container is surrounded by a border. The content of the area container can be reduced by clicking on the title - and resized by clicking again on the title.
- **ROWTABAREA and COLTABAREA**
  These are containers holding different pages (TABPAGE elements) which can be toggled.
- **ROWTABLE0 and COLTABLE0**
  These are containers you do not see; i.e. a container does not have any borders or any special coloring. Use it just for arranging elements inside the container.
- **ROWDYNAVIS and COLDYNAVIS**
  This is a container that is the same as the ROWTABLE0 or COLTABLE0 container but with an additional feature: You can control the visibility of the whole container dynamically by an adapter property. Use this container if you want to display or hide a certain area of your screen depending on some business logic.

  A typical example is an address management: the user enters an address located in the United States. Therefore, an additional area has to appear in which the user enters the state information. For other countries, this area is not required and should not be visible.

Row Containers

The containers have a row implementation and a column implementation.

Row containers occupy the whole available width they can obtain. They are placed directly in other containers. You can place several row containers inside one container. Therefore, they are arranged one below the other.

Example:

```
<pagebody>
    <rowarea name="Area 1">
    </rowarea>
    <rowarea name="Area 2">
    </rowarea>
    <rowarea name="Area 3">
    </rowarea>
</pagebody>
```

The above XML layout produces the following HTML page:



Column Containers

Column containers are placed inside rows, i.e. into TR rows or ITR rows. You can place several column containers inside one row. Therefore, they are arranged in a way that one column container follows the other horizontally.

Example:

```
<pagebody>
    <itr width="100%">
        <colarea name="Area 1" width="33%">
        </colarea>
        <hdist>
        </hdist>
        <colarea name="Area 2" width="33%">
        </colarea>
        <hdist>
        </hdist>
        <colarea name="Area 3" width="33%">
        </colarea>
    </itr>
</pagebody>
```

The above XML layout produces the following HTML page:



With column containers, you have to specify the width (either as a pixel value or as a percentage value) of the container. Note that - if using percentage widths - you have to place them into an ITR row that itself occupies the whole available width (`itr width="100%"`).

Row and Column Containers in Combination

It is possible to use row and column containers in combination. The following example combines the two examples shown above.

```
<pagebody>
    <rowarea name="Area1">
    </rowarea>
    <rowarea name="Area 2">
    </rowarea>
    <rowarea name="Area 3">
    </rowarea>
    <itr width="100%">
        <colarea name="Area 1" width="33%">
        </colarea>
        <hdist>
        </hdist>
        <colarea name="Area 2" width="33%">
        </colarea>
        <hdist>
        </hdist>
        <colarea name="Area 3" width="33%">
        </colarea>
    </itr>
</pagebody>
```

The HTML page looks as follows:

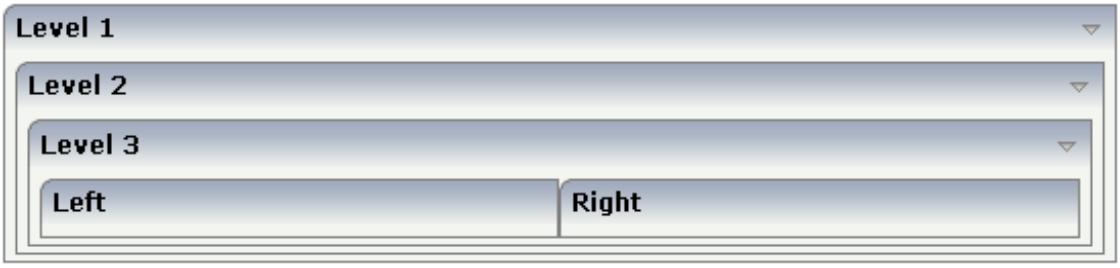


Nesting Containers

It is possible to nest containers - one into another - in any way. Example:

```
<pagebody>
    <rowarea name="Level 1">
        <rowarea name="Level 2">
            <rowarea name="Level 3">
                <itr width="100%">
                    <colarea name="Left" width="50%">
                    </colarea>
                    <hdist>
                    </hdist>
                    <colarea name="Right" width="50%">
                    </colarea>
                </itr>
            </rowarea>
        </rowarea>
    </rowarea>
</pagebody>
```

The above XML code produces the following HTML page:



11 ROWAREA and COLAREA

- ROWAREA Properties 68
- COLAREA Properties 75

The ROWAREA or COLAREA container represents an area surrounded by a border and which may have a title text. By clicking on the title of such a control, the inner content is hidden (the ROWAREA or COLAREA is “folded”).

ROWAREA Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Visibility			
foldable	The "folding"-function that is available by clicking on the title of the area can be switched off ("false"). "True" is the default.	Optional	true false
foldableprop	Name of the adapter parameter that dynamically controls whether clicking on the title of the area will fold/unfoald this area. Valid values provided by the adapter parameter are TRUE (=foldable) and FALSE(=not foldable).	Optional	
foldedprop	Name of adapter parameter which controls whether the content of the ROWAREA is folded (true) or displayed (false). By using this property you can dynamically control the "folded"-status of the control at runtime.	Optional	
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you	Optional	screen server

	want to pass one changed value to all its representation directly after changing the value.		
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Appearance			
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	
imageprop	Name of adapter parameter which dynamically provides the URL of the image that is shown inside the control. The URL must either be an absolute URL or a relative URL.	Optional	
withtoppadding	The control by default renders some blank vertical space (2 pixels) on top of its area. Reason: if you vertically arrange one ROW/COLAREA after the other then automatically some distance is put between. By specifying "false" you can avoid this behaviour. "	Optional	true false
withleftborder	The control normally renders a black border around its area. With the properties WITHLEFTBORDER, WITHRIGHTBORDER and WITHBOTTOMBORDER you can avoid this. Reason behind: sometimes you want a ROWAREA/COLAREA to be used as "neighbour" of other ROWAERA/COLAREA controls. In this case one of the "neighbours" has	Optional	true false

	to avoid the rendering of border lines - otherwise two border lines will be rendered.		
withtopborder	See description of WITHLEFTBORDER property.	Optional	true false
withrightborder	See description of WITHLEFTBORDER property.	Optional	true false
withbottomborder	See description of WITHLEFTBORDER property.	Optional	true false
paddingleft	Number of pixels between the left border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
paddingright	Number of pixels between the right border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
areastyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
contenttablestyle	CSS style definition that is applied to the content part of the ROWAREA control.	Optional	background-color: #FF0000

			color: #0000FF font-weight: bold
notabstop	The title of the area by default can be used by the user to hide/show the area's content. In order to also reach this title with the tab-key is part of the normal tab-sequence of a page. Set this property to "true" if you do not want to make the title reachable by tab-key. As consequence hiding/showing will only be available by mouse-clicking on the title.	Optional	true false
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10

			32767
withcontenttoppadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area. By specifying "false" you can avoid this behaviour.	Optional	true false
withcontentbottompadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area. By specifying "false" you can avoid this behaviour.	Optional	true false
withfadedtoggling	The animation of the controls can be switched off! Please take a look in your cisconfig.xml file. Set animatecontrols="true" (default) if you generally want to animate all of your controls. The rowarea control has a seperate switch (withfadedtoggling = true/false) to (de)activate the 'FadedToggling' animation especially for this single rowarea control. Notice: Entering true or false into the withfadedtoggling attribute overwrites the general animatecontrols setting !	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	
titlerowontop	Default value is 'true'. If set to 'false' the titlerow is rendered at the bottom of this area.	Optional	true false
toggleimgtitle	A text that is displayed as tooltip of the toggle image.	Optional	
toggleimgtitletextid	Multi language dependent text that is displayed as tooltip of the toggle image.	Optional	

	Do not specify a "toogleimagetitle" inside the control if specifying a "toggleimagetextid".		
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The	Optional	

	Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.		

COLAREA Properties

The properties of COLAREA are very similar to those of ROWAREA.

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.	Optional	true false
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	
imageprop	Name of adapter parameter which dynamically provides the URL of the image that is shown inside the control. The URL must either be an absolute URL or a relative URL.	Optional	
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control.	Optional	true false

	If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.		
withleftborder	The control normally renders a black border around its area. With the properties WITHLEFTBORDER, WITHRIGHTBORDER and WITHBOTTOMBORDER you can avoid this. Reason behind: sometimes you want a ROWAREA/COLAREA to be used as "neighbour" of other ROWAERA/COLAREA controls. In this case one of the "neighbours" has to avoid the rendering of border lines - otherwise two border lines will be rendered.	Optional	true false
withtopborder	See description of WITHLEFTBORDER property.	Optional	true false
withrightborder	See description of WITHLEFTBORDER property.	Optional	true false
withbottomborder	See description of WITHLEFTBORDER property.	Optional	true false
paddingleft	Number of pixels between the left border and the area's content. Default is 5 pixels.	Optional	1 2 3

			int-value
paddingright	Number of pixels between the right border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
areastyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
contenttablestyle	CSS style that is applied to the content are of the COLAREA control.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
withcontenttoppadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area. By specifying "false" you can avoid this behaviour.	Optional	true false
withcontentbottompadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area. By specifying "false" you can avoid this behaviour.	Optional	true false
titlerowontop	Default value is 'true'. If set to 'false' the titlerow is rendered at the bottom of this area.	Optional	true false

stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	
withtoppadding	The control by default renders some blank vertical space (2 pixels) on top of its area. Reason: if you vertically arrange one ROW/COLAREA after the other then automatically some distance is put between. By specifying "false" you can avoid this behaviour. "	Optional	true false
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

12 ROWAREAWITHHEADER

- Simple Example 82
- Right-to-left (RTL) Mode 83
- ROWAREAWITHHEADER Properties 83
- ROWAREAHEADER Properties 86
- ROWAREABODY Properties 87

This container represents an area surrounded by a border which may have a title text. By clicking on the title, the inner content is hidden (the container is “folded”). You can place icons (**ICON**, **ICONLIST**) into the header line (ROWAREAHEADER). Other content is placed into the ROWAREABODY container.

Simple Example

```
<rowareawithheader>
    <rowareaheader name="Note">
        <hdist width="20">
        </hdist>
        <icon image="../HTMLBasedGUI/images/cut.gif" method="onCut">
        </icon>
        <hdist width="6">
        </hdist>
        <icon image="../HTMLBasedGUI/images/copy.gif" method="onCopy">
        </icon>
        <hdist width="6">
        </hdist>
        <icon image="../HTMLBasedGUI/images/paste.gif" method="onPaste">
        </icon>
    </rowareaheader>
    <rowareabody>
        <itr takefullwidth="true">
            <text valueprop="text" width="100%" rows="5">
            </text>
        </itr>
    </rowareabody>
</rowareawithheader>
```

The above XML layout produces a page which looks as follows:



There are three icons within the header line (ROWAREAHEADER). The text box is placed into the body container (ROWAREABODY).

Right-to-left (RTL) Mode

To properly support right-to-left (RTL) mode, it is required to set a foldableprop for correct display of right-to-left content.

For example: `<rowareawithheader foldableprop="myfoldableprop">`.

ROWAREAWITHHEADER Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Visibility			
foldable	The "folding"-function that is available by clicking on the title of the area can be switched off ("false"). "True" is the default.	Optional	true false
foldableprop	Name of the adapter parameter that dynamically controls whether clicking on the title of the area will fold/unfoald this area. Valid values provided by the adapter parameter are TRUE (=foldable) and FALSE(=not foldable).	Optional	
foldedprop	Name of adapter parameter which controls whether the content of the ROWAREA is folded (true) or displayed (false).	Optional	

	By using this property you can dynamically control the "folded"-status of the control at runtime.		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Appearance			
height	(already explained above)		
withleftborder	The control normally renders a black border around its area. With the properties WITHLEFTBORDER, WITHRIGHTBORDER and WITHBOTTOMBORDER you can avoid this. Reason behind: sometimes you want a ROWAREA/COLAREA to be used as "neighbour" of other ROWAERA/COLAREA controls. In this case one of the "neighbours" has to avoid the rendering of border lines - otherwise two border lines will be rendered.	Optional	true false
withtopborder	See description of WITHLEFTBORDER property.	Optional	true false
withrightborder	See description of WITHLEFTBORDER property.	Optional	true false
withbottomborder	See description of WITHLEFTBORDER property.	Optional	true

			false
withtoppadding	The control by default renders some blank vertical space (2 pixels) on top of its area. Reason: if you vertically arrange one ROW/COLAREA after the other then automatically some distance is put between. By specifying "false" you can avoid this behaviour. "	Optional	true false
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	
imageprop	Name of adapter parameter which dynamically provides the URL of the image that is shown inside the control. The URL must either be an absolute URL or a relative URL.	Optional	
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Optional	
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.	Optional	true false
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name	Optional	

	is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

ROWAREAHEADER Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Appearance			
align	Horizontal alignment of the controls inside the header line.	Optional	left

			center right
notabstop	The title of the area by default can be used by the user to hide/show the area's content. In order to also reach this title with the tab-key is part of the normal tab-sequence of a page. Set this property to "true" if you do not want to make the title reachable by tab-key. As consequence hiding/showing will only be available by mouse-clicking on the title.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767

ROWAREABODY Properties

Basic			
paddingleft	Number of pixels between the left border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
paddingright	Number of pixels between the right border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
bodystyle	CSS style definition that is directly passed into this control.	Optional	background-color: #FF0000 color: #0000FF

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		font-weight: bold
withcontenttoppadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area. By specifying "false" you can avoid this behaviour.	Optional	true false
withcontentbottompadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area. By specifying "false" you can avoid this behaviour.	Optional	true false

13 ROWTABAREA and COLTABAREA

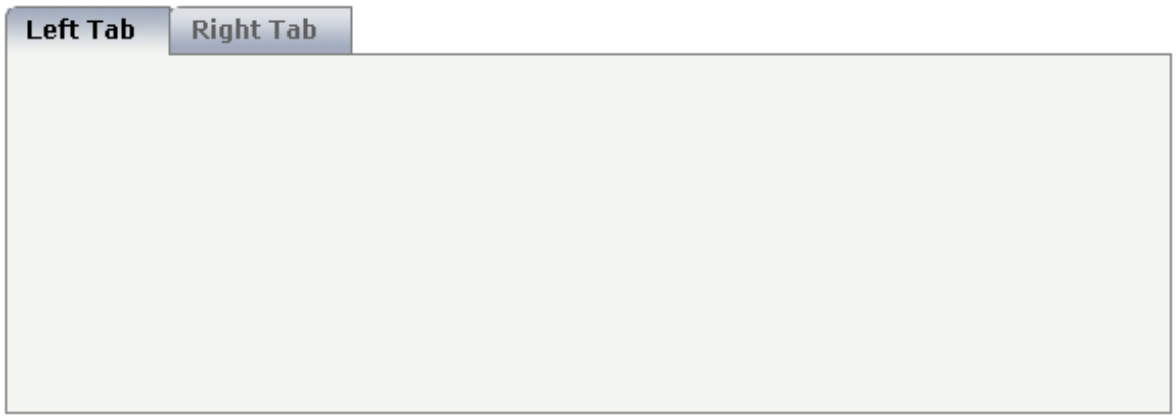
- ROWTABAREA Properties 91
- COLTABAREA Properties 117
- TABPAGE Properties 140
- The Most Common Error 141
- Example: Controlling which Tab is displayed by the Server Adapter 141
- Example: Controlling the Visibility of Tab Pages 143

The ROWTABAREA or COLTABAREA container is the representation of a tab control. A tab area consists of the ROWTABAREA or COLTABAREA definition. Inside this definition, you define TABPAGE containers representing the individual pages between which you can navigate.

Example:

```
<pagebody>
    <rowtabarea height="200" name1="Left Tab" page1="LEFT" name2="Right Tab" ↩
page2="RIGHT">
        <tabpage id="LEFT" takefullheight="true">
        </tabpage>
        <tabpage id="RIGHT" takefullheight="true">
        </tabpage>
    </rowtabarea>
</pagebody>
```

The above XML layout produces the following page:



Inside the ROWTABAREA definition, specify the name and the ID of each area you want to display. Pay attention to the naming of the `page*` properties: the name must not contain any blank spaces or non-alphanumeric characeters. Start the `page*` values with a character, not with a number.

Specify the individual toggle areas - by the TABPAGE definition. Each TABPAGE holds an ID which must be equal to the definition on ROWTABAREA level. Each TABPAGE has a `display` property which is set to "none" for all TABPAGE definitions except the first one.

Each TABPAGE is a container itself - i.e. inside the TABPAGE, place controls (or containers) by adding ITR or TR rows and place elements into these rows.

ROWTABAREA Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%
leftindent	Inserts a horizontal distance left of the first "tab" and shifts the "tabs" to the right as consequence. The value you may define represents the number of pixels that are inserted.	Optional	1 2 3 int-value
scrollable	If set to "true" then small icons will appear on the right border of the control. If the size of the "tabs" is too big and some tabs are cut as consequence then you can use these icons for scrolling left and right.	Optional	true false
name1	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Sometimes obligatory	
textid1	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Sometimes obligatory	

page1	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Obligatory	
withclose1	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name2	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid2	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page2	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose2	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name3	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid3	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page3	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below -	Optional	

	holding exactly the id that is defined in the PAGE property.		
withclose3	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name4	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid4	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page4	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose4	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name5	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid5	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page5	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose5	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false

name6	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid6	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page6	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose6	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name7	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid7	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page7	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose7	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name8	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid8	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	

page8	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose8	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name9	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid9	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page9	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose9	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name10	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid10	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page10	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below -	Optional	

	holding exactly the id that is defined in the PAGE property.		
withclose10	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name11	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid11	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page11	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose11	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name12	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid12	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page12	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose12	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false

name13	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid13	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page13	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose13	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name14	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid14	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page14	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose14	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name15	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid15	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	

page15	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose15	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
name16	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid16	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page16	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
withclose16	If you want a close-icon to be shown at the right top corner of the tab, set this property value to "true". Default is "false".	Optional	true false
Binding			
openedindexprop	Name of adapter parameter which represents the index of the "tab" that is currently opened. There are two ways of using the property: either you can define which "tab" should be opened or you can react to "tab" selections by the user. (Also have a look onto the property OPENMETHOD!). The property must be of type "int" or "Integer" (or "String"). The left most "tab" represents index "0", the next one "1", etc.	Optional	

openmethod	Name of the event that is sent to the adapter when the user does a "tab" selection. The index of the "tab" that is opened can be transferred to the adapter by using the property OPENEDINDEXPROP.	Optional	
visibleprop1	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop2	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop3	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop4	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop5	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop6	Name of property that defines if the corresponding tag is visible or not. NOTICE: If	Optional	

	you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.		
visibleprop7	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop8	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop9	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop10	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop11	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	

visibleprop12	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop13	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop14	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop15	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop16	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
disabledprop1	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	

disabledprop2	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop3	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop4	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop5	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop6	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop7	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop8	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop9	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In	Optional	

	COLTABAREA controls this property is only supported for IE.		
disabledprop10	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop11	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop12	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop13	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop14	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop15	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop16	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
titleprop1	Property of adapter that dynamically defines the title of the control. The title is displayed as tool	Optional	

	tip when the user moves the mouse onto the control.		
titleprop2	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop3	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop4	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop5	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop6	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop7	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop8	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop9	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop10	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop11	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop12	Property of adapter that dynamically defines the title of the control. The title is displayed as tool	Optional	

	tip when the user moves the mouse onto the control.		
titleprop13	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop14	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop15	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop16	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
tabselectedstyleprop1	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop1	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop1	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop2	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop2	Name of the adapter parameter that dynamically defines the style for a tab which is not selected	Optional	background-color: #FF0000

	and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.		color: #0000FF font-weight: bold
tabdisabledstyleprop2	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop3	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop3	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop3	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop4	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop4	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop4	Name of the adapter parameter that dynamically defines the style for a tab which is disabled.	Optional	background-color: #FF0000

	NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.		color: #0000FF font-weight: bold
tabselectedstyleprop5	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop5	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop5	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop6	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop6	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop6	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop7	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must	Optional	background-color: #FF0000 color: #0000FF

	also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.		font-weight: bold
tabunselectedstyleprop7	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop7	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop8	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop8	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop8	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop9	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop9	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property	Optional	background-color: #FF0000 color: #0000FF

	OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.		font-weight: bold
tabdisabledstyleprop9	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop10	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop10	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop10	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop11	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop11	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop11	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You	Optional	background-color: #FF0000 color: #0000FF

	don't have to set a value at runtime, but you need to specify a valid name.		font-weight: bold
tabselectedstyleprop12	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop12	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop12	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop13	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop13	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop13	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop14	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	don't have to set a value at runtime, but you need to specify a valid name.		
tabunselectedstyleprop14	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop14	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop15	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop15	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop15	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop16	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop16	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	value at runtime, but you need to specify a valid name.		
tabdisabledstyleprop16	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Appearance			
withleftborder	If specified as "false" then no left border will be drawn.	Optional	true false
withrightborder	If specified as "false" then no right border will be drawn.	Optional	true false
withbottomborder	If specified as "false" then no bottom border will be drawn.	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767

withtoppadding	The control by default renders some blank vertical space (2 pixels) on top of its area. Reason: if you vertically arrange one ROW/COLAREA after the other then automatically some distance is put between. By specifying "false" you can avoid this behaviour. "	Optional	true false
tabpagepaddingleft	Number of pixels between the left border and the area's content. Default is 5 pixels.	Sometimes obligatory	1 2 3 int-value
tabpagepaddingright	Number of pixels between the right border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
tabpagepaddingtop	Number of pixels between the top border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
tabpagepaddingbottom	Number of pixels between the bottom border and the area's content. Default is 5 pixels.	Optional	1 2 3 int-value
withflash	Adds animation effects when the user uses the control.	Optional	
Online Help			
title1	Tooltip text that appears on the corresponding tab.	Optional	
title2	Tooltip text that appears on the corresponding tab.	Optional	
title3	Tooltip text that appears on the corresponding tab.	Optional	

title4	Tooltip text that appears on the corresponding tab.	Optional	
title5	Tooltip text that appears on the corresponding tab.	Optional	
title6	Tooltip text that appears on the corresponding tab.	Optional	
title7	Tooltip text that appears on the corresponding tab.	Optional	
title8	Tooltip text that appears on the corresponding tab.	Optional	
title9	Tooltip text that appears on the corresponding tab.	Optional	
title10	Tooltip text that appears on the corresponding tab.	Optional	
title11	Tooltip text that appears on the corresponding tab.	Optional	
title12	Tooltip text that appears on the corresponding tab.	Optional	
title13	Tooltip text that appears on the corresponding tab.	Optional	
title14	Tooltip text that appears on the corresponding tab.	Optional	
title15	Tooltip text that appears on the corresponding tab.	Optional	
title16	Tooltip text that appears on the corresponding tab.	Optional	
titletextid1	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid2	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid3	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid4	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	

titletextid5	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid6	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid7	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid8	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid9	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid10	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid11	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid12	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid13	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid14	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid15	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	

titletextid16	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
Comment			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Miscellaneous			
testtoolid1	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid2	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid3	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid4	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid5	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid6	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid7	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid8	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid9	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid10	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid11	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

testtoolid12	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid13	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid14	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid15	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
testtoolid16	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

COLTABAREA Properties

The properties of COLTABAREA are very similar to those of ROWTABAREA.

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
leftindent	Inserts a horizontal distance left of the first "tab" and shifts the "tabs" to the right as consequence.	Optional	1

	The value you may define represents the number of pixels that are inserted.		2 3 int-value
scrollable	If set to "true" then small icons will appear on the right border of the control. If the size of the "tabs" is too big and some tabs are cut as consequence then you can use these icons for scrolling left and right.	Optional	true false
name1	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Sometimes obligatory	
textid1	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Sometimes obligatory	
page1	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Obligatory	
name2	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid2	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page2	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name3	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	

textid3	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page3	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name4	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid4	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page4	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name5	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid5	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page5	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	

name6	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid6	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page6	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name7	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid7	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page7	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name8	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid8	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page8	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below -	Optional	

	holding exactly the id that is defined in the PAGE property.		
name9	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid9	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page9	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name10	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid10	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page10	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name11	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid11	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page11	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters.	Optional	

	For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.		
name12	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid12	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page12	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name13	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid13	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page13	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name14	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid14	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page14	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters.	Optional	

	For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.		
name15	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid15	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page15	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
name16	Text that is shown in the corresponding "tab". Either define the text as NAME or as language dependent TEXTID.	Optional	
textid16	Text ID that is transferred in a corresponding literal at runtime by the multi language management.	Optional	
page16	Id of the TABPAGE that is defined as child of the TABAREA. Use an id that is unique within the page and that is a "healthy" id: starting with characters, without blanks and without "strange" characters. For each "tab" of the TABAREA you have to create one corresponding TABPAGE below - holding exactly the id that is defined in the PAGE property.	Optional	
Binding			
openedindexprop	Name of adapter parameter which represents the index of the "tab" that is currently opened. There are two ways of using the property: either you can define which "tab" should be opened or you can react to "tab" selections by the user. (Also have a look onto the property OPENMETHOD!).	Optional	

	The property must be of type "int" or "Integer" (or "String"). The left most "tab" represents index "0", the next one "1", etc.		
openmethod	Name of the event that is sent to the adapter when the user does a "tab" selection. The index of the "tab" that is opened can be transferred to the adapter by using the property OPENEDINDEXPROP.	Optional	
visibleprop1	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop2	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop3	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop4	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop5	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a	Optional	

	value at runtime, but you need to specify a valid name.		
visibleprop6	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop7	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop8	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop9	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop10	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop11	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute	Optional	

	OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.		
visibleprop12	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop13	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop14	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop15	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
visibleprop16	Name of property that defines if the corresponding tag is visible or not. NOTICE: If you want the framework to automatically set the focus to the first visible tab you also must apply a name for the attribute OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	
disabledprop1	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In	Optional	

	COLTABAREA controls this property is only supported for IE.		
disabledprop2	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop3	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop4	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop5	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop6	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop7	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop8	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop9	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at	Optional	

	runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.		
disabledprop10	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop11	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop12	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop13	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop14	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop15	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	
disabledprop16	Name of the adapter parameter that dynamically defines if the control is disabled or enabled at runtime. If the value at runtime is set to TRUE the control is visible but disabled. In COLTABAREA controls this property is only supported for IE.	Optional	

titleprop1	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop2	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop3	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop4	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop5	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop6	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop7	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop8	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop9	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop10	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop11	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	

titleprop12	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop13	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop14	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop15	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
titleprop16	Property of adapter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
tabselectedstyleprop1	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop1	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop1	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop2	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabunselectedstyleprop2	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop2	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop3	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop3	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop3	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop4	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop4	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabdisabledstyleprop4	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop5	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop5	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop5	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop6	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop6	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop6	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabselectedstyleprop7	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop7	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop7	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop8	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop8	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop8	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop9	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabunselectedstyleprop9	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop9	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop10	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop10	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop10	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop11	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop11	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabdisabledstyleprop11	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop12	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop12	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop12	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop13	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop13	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop13	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabselectedstyleprop14	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop14	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop14	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop15	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabunselectedstyleprop15	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop15	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabselectedstyleprop16	Name of the adapter parameter that dynamically defines the style for a tab which is selected. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

tabunselectedstyleprop16	Name of the adapter parameter that dynamically defines the style for a tab which is not selected and not disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabdisabledstyleprop16	Name of the adapter parameter that dynamically defines the style for a tab which is disabled. NOTICE: When using this property you must also set the property OPENEDINDEXPROP. You don't have to set a value at runtime, but you need to specify a valid name.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Appearance			
withleftborder	If specified as "false" then no left border will be drawn.	Optional	
withrightborder	If specified as "false" then no right border will be drawn.	Optional	
withbottomborder	If specified as "false" then no bottom border will be drawn.	Optional	
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Online Help			

title1	Tooltip text that appears on the corresponding tab.	Optional	
title2	Tooltip text that appears on the corresponding tab.	Optional	
title3	Tooltip text that appears on the corresponding tab.	Optional	
title4	Tooltip text that appears on the corresponding tab.	Optional	
title5	Tooltip text that appears on the corresponding tab.	Optional	
title6	Tooltip text that appears on the corresponding tab.	Optional	
title7	Tooltip text that appears on the corresponding tab.	Optional	
title8	Tooltip text that appears on the corresponding tab.	Optional	
title9	Tooltip text that appears on the corresponding tab.	Optional	
title10	Tooltip text that appears on the corresponding tab.	Optional	
title11	Tooltip text that appears on the corresponding tab.	Optional	
title12	Tooltip text that appears on the corresponding tab.	Optional	
title13	Tooltip text that appears on the corresponding tab.	Optional	
title14	Tooltip text that appears on the corresponding tab.	Optional	
title15	Tooltip text that appears on the corresponding tab.	Optional	
title16	Tooltip text that appears on the corresponding tab.	Optional	
titletextid1	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid2	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid3	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management	Optional	

	replaces the textid with a language dependent literal.		
titletextid4	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid5	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid6	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid7	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid8	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid9	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid10	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid11	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid12	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid13	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid14	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management	Optional	

	replaces the textid with a language dependent literal.		
titletextid15	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
titletextid16	Text ID for the tooltip of the corresponding "tab". At runtime the multi language management replaces the textid with a language dependent literal.	Optional	
Comment			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

TABPAGE Properties

Basic			
id	Id of the TABPAGE. Each page has an id that refers to the PAGE1 .. PAGE9 definition inside the ROW/COLTABAREA control that contains the TABPAGE. Clicking a "tab" will display the TABPAGE with the associated id.	Obligatory	
display	Initial display status of the TABPAGE. The first TABPAGE inside the ROW/COLTABAREA control must be set to "". All others need to be set to "none". - If a ROW/COLTABAREA should show up with two or more pages being visible one below the other then check the setting of this property!"	Sometimes obligatory	
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.	Optional	true false
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control.	Optional	true false

	If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

The Most Common Error

Do you receive errors when clicking in the tabs? Then take a further look at the ID assignments in the ROWTABAREA or COLTABAREA control on the one hand, and in the TABPAGE control on the other hand: each `page*` property of a ROWTABAREA or COLTABAREA defines an ID that must exactly match an `id` property of TABPAGE.

If you have more than one ROWTABAREA or COLTABAREA inside your page: do not use the same IDs - each ID must be unique throughout one page.

Example: Controlling which Tab is displayed by the Server Adapter

The following example demonstrates the usage of the property `openedindexprop` on ROWTAB-AREA level:



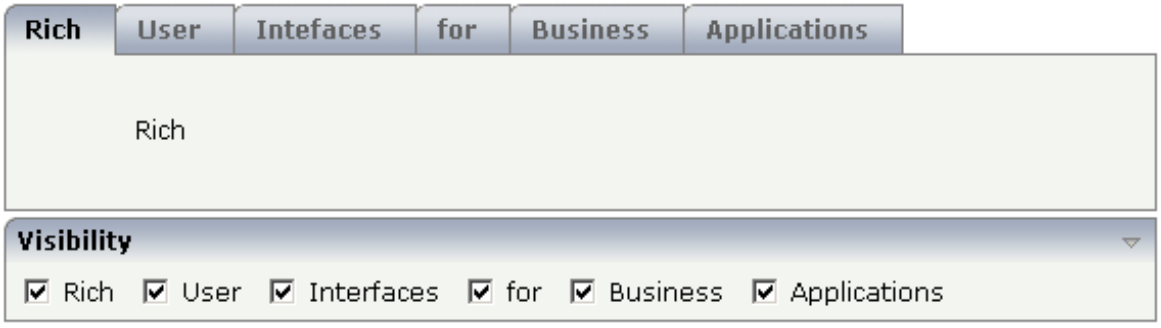
The user selects the value of the property `index` using the combo control. The `index` property controls also which tab is displayed inside the ROWTABAREA control.

The layout definition is as follows:

```
<pagebody>
    <rowarea name="Dynamic setting of index in TABAREA">
        <itr>
            <label name="Index" width="100">
            </label>
            <combofix valueprop="index" size="1" flush="server">
                <combooption name="First  (=0)" value="0">
                </combooption>
                <combooption name="Second  (=1)" value="1">
                </combooption>
                <combooption name="Third (=2)" value="2">
                </combooption>
            </combofix>
        </itr>
    </rowarea>
    <rowtabarea height="200" openedindexprop="index"
                name1="First" page1="FIRST"
                name2="Second" page2="SECOND"
                name3="Third" page3="THIRD">
        <tabpage id="FIRST">
        </tabpage>
        <tabpage id="SECOND">
        </tabpage>
        <tabpage id="THIRD">
        </tabpage>
    </rowtabarea>
</pagebody>
```

Example: Controlling the Visibility of Tab Pages

For each individual tab page, you can control at runtime whether it is visible or not. The following example allows the user to control the visibility of tabs using check boxes:



The XML layout is:

```
<rowtabarea height="100" name1="Rich" page1="RICH" visibleprop1="page1Visibility"
                         name2="User" page2="USER" visibleprop2="page2Visibility"
                         name3="Intefaces" page3="INTERFACES" ↩
visibleprop3="page3Visibility"
                         name4="for" page4="FOR" visibleprop4="page4Visibility"
                         name5="Business" page5="BUSINESS" ↩
visibleprop5="page5Visibility"
                         name6="Applications" page6="APPLICATIONS"
                                              visibleprop6="page6Visibility">
    <tabpage id="RICH">
        <vdist height="20">
        </vdist>
        <itr>
            <hdist width="60">
            </hdist>
            <label name="Rich" asplaintext="true" textalign="center">
            </label>
        </itr>
    </tabpage>
    <tabpage id="USER">
        ...
    </tabpage>
    ...
    ...
    ...
<rowarea name="Visibility">
    <itr>
        <checkbox valueprop="page1Visibility" flush="server">
        </checkbox>
        <hdist>
```

```
        </hdist>
        <label name="Rich" asplaintext="true">
        </label>
        <hdist width="10">
        </hdist>
        <checkbox valueprop="page2Visibility" flush="server">
        </checkbox>
        <hdist>
        </hdist>
        <label name="User" asplaintext="true">
        </label>
        <hdist width="10">
        </hdist>
        <checkbox valueprop="page3Visibility" flush="server">
        </checkbox>
        <hdist>
        </hdist>
        <label name="Interfaces" asplaintext="true">
        </label>
        <hdist width="10">
        </hdist>
        <checkbox valueprop="page4Visibility" flush="server">
        </checkbox>
        <hdist>
        </hdist>
        <label name="for" asplaintext="true">
        </label>
        <hdist width="10">
        </hdist>
        <checkbox valueprop="page5Visibility" flush="server">
        </checkbox>
        <hdist>
        </hdist>
        <label name="Business" asplaintext="true">
        </label>
        <hdist width="10">
        </hdist>
        <checkbox valueprop="page6Visibility" flush="server">
        </checkbox>
        <hdist>
        </hdist>
        <label name="Applications" asplaintext="true">
        </label>
        <hdist width="10">
        </hdist>
    </itr>
</rowarea>
```

You see that the definition of the properties that control the visibility of tab pages is done in the ROWTABAREA (not on TABPAGE level). The check boxes reference the same adapter properties as used on ROWTABAREA level.

Note: In the previous example, the `openedindexprop` property of the ROWTABAREA was used. Be aware of the fact that each tab page still keeps its stable index position - no matter whether it is displayed or not.

14 ROWTABLE0 and COLTABLE0

- ROWTABLE0 Properties 149
- COLTABLE0 Properties 151

The ROWTABLE0 or COLTABLE0 container is not visible. Normally, it is just used for arranging controls. The following example shows how to define two columns - inside a ROWAREA - to arrange controls:

```
<pagebody>
    <rowarea name="Area 1">
        <itr takefullwidth="true">
            <coltable0 width="50%" takefullheight="true">
                <itr>
                    <label name="Factor 1" width="100">
                    </label>
                    <field valueprop="factor1" length="5">
                    </field>
                </itr>
            </coltable0>
            <coltable0 width="50%" takefullheight="true">
                <itr>
                    <label name="Factor 2" width="100">
                    </label>
                    <field valueprop="factor2" length="5">
                    </field>
                </itr>
            </coltable0>
        </itr>
    </rowarea>
</pagebody>
```

The result looks as follows:



Inside the ROWAREA, two COLTABLE0 tags are placed - each occupying 50% of the width. Each COLTABLE0 area builds - independently from the other - its own table rows (ITR rows in the example).

All complex field arrangements should be done by using ROWTABLE0/COLTABLE0 tags as shown in the example.

ROWTABLE0 Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
align	Alignment of the content of the ITR row. Background: the ITR as independent table row renders a table into its content area. Inside this table a row is opened in which the controls are placed. This table normally is starting on the left of the ITR row. With this ALIGN property you can explicitly define the alignment of the table.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut.	Optional	true false

	You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.		
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
flashprop	Name of the adapter parameter that triggers a "flashing" of the area. "Flashing" means that the area is animated for a short point of time in order to make the user aware that e.g. some change of data happened inside the area. The value is an index - whenever you change the index then a flashing of the control is triggered on client side. Pay attention: do not mix the "flashing" of an area with the "flushing" of controls - "flushing" is the way an input control (e.g. field) triggers server side updates when the user changed the value, "flashing" is pure animation.	Optional	

COLTABLE0 Properties

The properties for COLTABLE0 are very similar to those of ROWTABLE0.

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
widthprop	Name of adapter parameter that dynamically defines the height of the control. Must return a valid width.	Optional	
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.	Optional	true false
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as	Optional	true false

	normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.		
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

15 ROWDYNAVIS and COLDYNAVIS

- ROWDYNAVIS Properties 155
- COLDYNAVIS Properties 157
- Some Comments on Controlling the Visibility of Controls 159

The ROWDYNAVIS or COLDYNAVIS container is used to add dynamic reaction to your layout.

The container is not visible - similar to the TABLE0 container. What is the difference? You control the appearance of the container by an adapter property. Have a look at the following example.



If you enter "United States" as a country, the input line for the state will appear under the input line for the country:



The XML code looks as follows:

```
<rowarea name="Address Input">
    <itr>
        <label name="Country" width="100">
        </label>
        <field valueprop="country" flush="true" length="30">
        </field>
    </itr>
    <rowdynavis valueprop="visible">
        <itr>
            <label name="State" width="100">
            </label>
            <field valueprop="state" length="30">
            </field>
        </itr>
    </rowdynavis>
</rowarea>
```

A ROWDYNAVIS container is placed inside the ROWAREA container.

ROWDYNAVIS Properties

Basic			
valueprop	Name of adapter parameter that defines if the area is visible ("true") or invisible ("false").	Obligatory	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it	Optional	true

	requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.		false
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

COLDYNAVIS Properties

The properties of COLDYNAVIS are very similar to those of ROWDYNAVIS.

Basic			
valueprop	Name of adapter parameter that defines if the area is visible ("true") or invisible ("false").	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.	Optional	true false
style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:	Optional	background-color: #FF0000 color: #0000FF

	border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		font-weight: bold
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and	Optional	

	#GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

Some Comments on Controlling the Visibility of Controls

ROWDYNAVIS and COLDYNAVIS are container controls that are explicitly defined to provide an area which can be explicitly switched on and off. In addition you will later on see that many controls can control their visiblity and their input status by themselves. For example, a FIELD control can specify if it is invisible, editable, holding an error input etc. in a dynamic way. You may also have noticed that an ITR row definition has an associated `visibleprop` property - linking to a data property that dynamically controls the visibility of the row at runtime.

Use ROWDYNAVIS and COLDYNAVIS for explicitly defining container areas to be switched on/off. Use the control's binding to properties to do the fine-granular control of visibility inside one container.

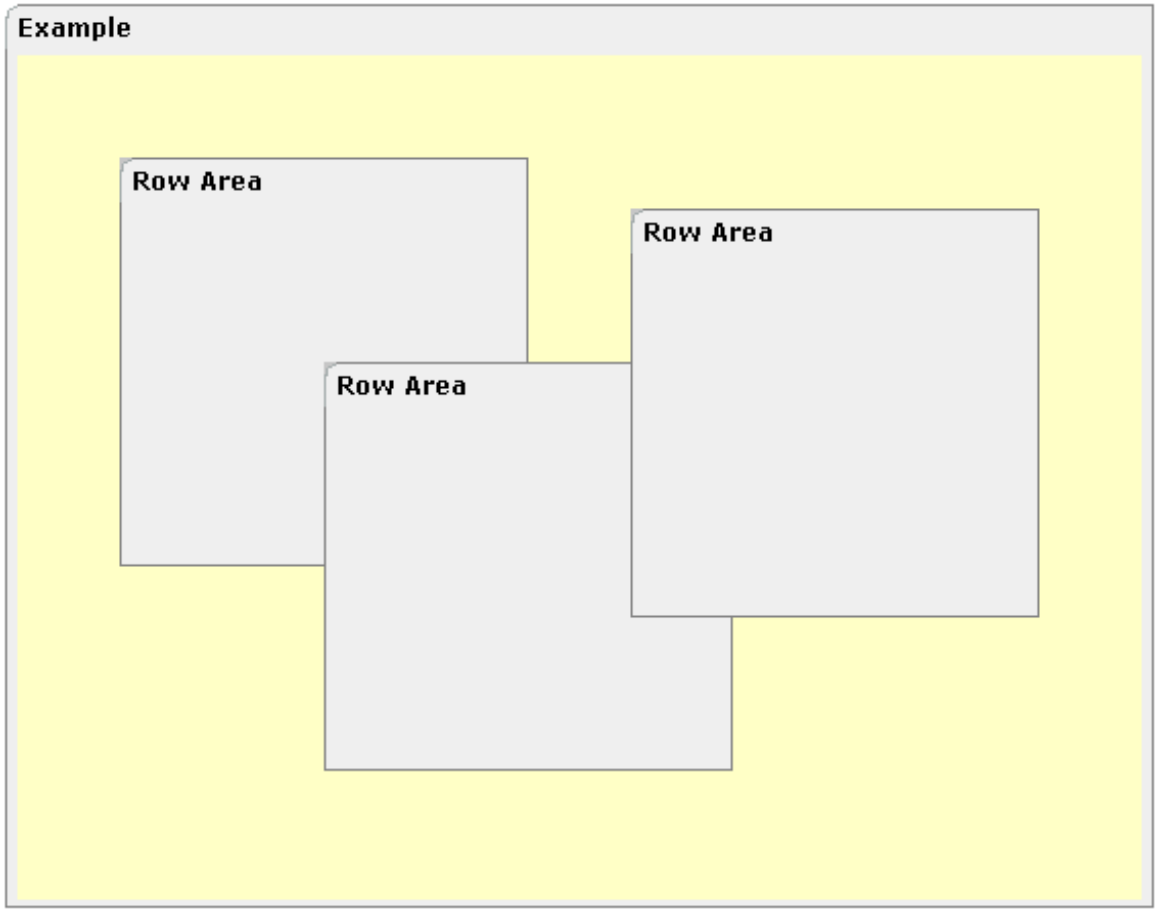
A bad example of usage would be if you place a COLDYNAVIS container around each FIELD that you want to control in means of visibility. Use the FIELD's `statusprop` property instead.

16 ROWDIV and INNERDIV

- When to Use ROWDIV and INNERDIV Containers 164
- ROWDIV Properties 164
- INNERDIV Properties 165

The ROWDIV container represents an area with a defined size. Inside this area you can arrange INNERDIV containers. The INNERDIV containers have a defined x-, y- and z-position inside the ROWDIV area, and they have a defined width and height. INNDERDIV containers can overlap; by using the z-position, you can define which INNERDIV container is on top of which other INNERDIV container. Inside an INNERDIV container, you can arrange any other container or control - just as with normal containers.

Have a look at the following example:

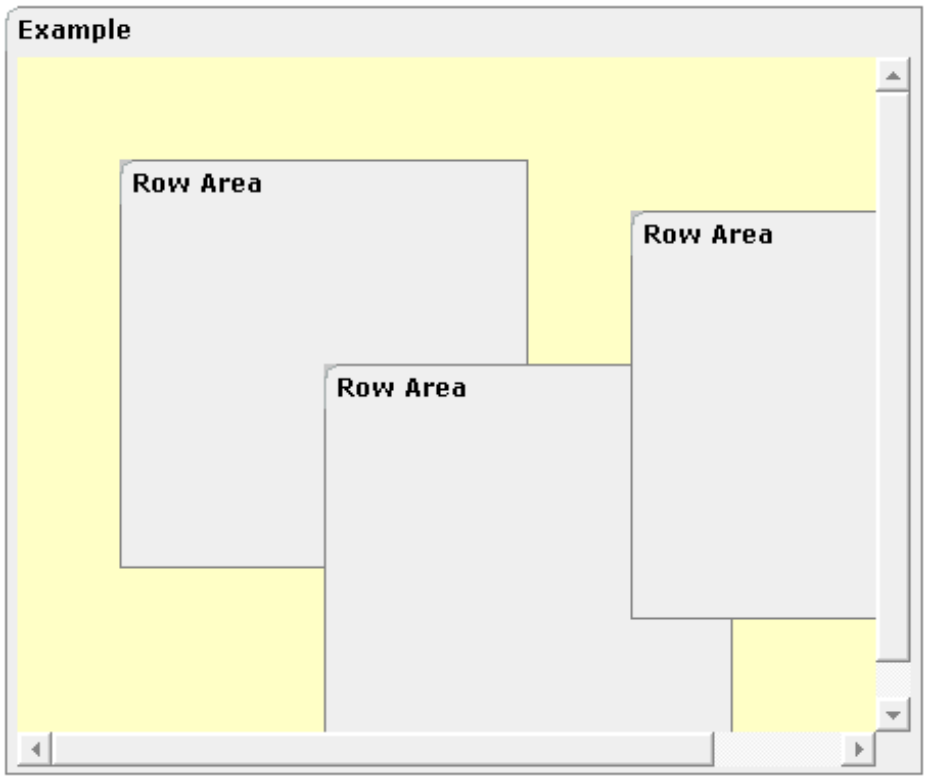


Inside a ROWAREA container, a ROWDIV container is arranged. Inside the ROWDIV container, three INNERDIV containers are arranged - each one holding a ROWAREA.

The XML layout definition looks as follows:

```
<rowarea name="Example" height="100%">
    <rowdiv height="100%" style="background-color: #FFFFC0">
        <innerdiv width="200" height="200" zindex="99" left="150" top="150"
                  style="background-color: #C0C0C0">
            <rowarea name="Row Area" height="100%" withtoppadding="false">
            </rowarea>
        </innerdiv>
        <innerdiv width="200" height="200" zindex="98" left="50" top="50"
                  style="background-color: #C0C0C0">
            <rowarea name="Row Area" height="100%" withleftborder="true" ↩
withtopborder="true"
                     withrightborder="true" withbottomborder="true" ↩
withtoppadding="false">
            </rowarea>
        </innerdiv>
        <innerdiv width="200" height="200" zindex="100" left="300" top="75"
                  style="background-color: #C0C0C0">
            <rowarea name="Row Area" height="100%" withtoppadding="false">
            </rowarea>
        </innerdiv>
    </rowdiv>
</rowarea>
```

If the ROWDIV area is too small to hold the INNERDIV containers, then the ROWDIV area starts scrolling:



When to Use ROWDIV and INNERDIV Containers

The typical usage scenarios of ROWDIV and INNERDIV containers is:

- when you want to place a certain area at a certain position on the screen - without wanting to explicitly define VDIST/HDIST elements;
- when you want to explicitly work with overlapping areas.

Note that the parallel usage of pixel and percentage sizing is not supported with ROWDIV and INNERDIV in the same way as supported with normal containers (for example, ROWAREA and COLAREA). With normal containers, you can specify scenarios like the following: the left container occupies 200 pixels, the right container occupies 100%. The table rendering is clever enough to render the result accordingly. With INNERDIV containers, the percentage definitions are always in relation to the height and width of the surrounding ROWDIV control.

Consequence: Do not use ROWDIV and INNERDIV for the basic structuring of containers inside your page, but only use them for the two usage aspects mentioned before.

ROWDIV Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%
style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000	Optional	

	background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
divclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

INNERDIV Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20").	Obligatory	100 150 200 250 300

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 400 50% 100%
left	Left position of control. Either define a pixel value ("100") or a percentage value ("30%").	Obligatory	
top	Top position of control. Either define a pixel value ("100") or a percentage value ("30%").	Obligatory	
zindex	Z-index of the control. If two controls overlap then the one with the higher z-index is drawn in front of the other one.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
leftdistance	If set to "true" then a small distance (3px) is kept between the left border of the control and its content. Default is "false".	Optional	true false
rightdistance	If set to "true" then a small distance (3px) is kept between the right border of the control and its content. Default is "false".	Optional	true false
style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
Binding			
widthprop	Name of adapter parameter that dynamically provides the width of the control. Must return a valid width.	Optional	

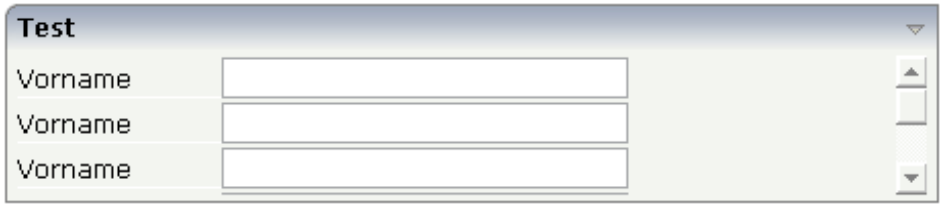
leftprop	Name of adapter parameter that dynamically provides the left position of the control. Must return a valid value for 'left position'.	Optional	
dropwidthprop	Name of the adapter parameter that dynamically provides the width of the drop target.	Optional	
dropoffsetprop	Name of the adapter parameter that dynamically provides the offset used for the drop target.	Optional	
dropmethod	Event which is triggered in the Natural program when the drop operation is finished.	Optional	

17 ROWSCROLLAREA

- ROWSCROLLAREA Properties 171

The ROWSCROLLAREA represents a container area with a certain size. The container is not visible. If the contents of the container area exceed the size of the container area, then scroll bars are added accordingly.

Have a look at the following example:



Inside a normal ROWAREA with the title "Test", a ROWSCROLLAREA is positioned. Inside the ROWSCROLLAREA, a number of lines is arranged so that the total height of the lines exceeds the height of the ROWSCROLLAREA. Consequently, a vertical scroll bar is shown on the right.

The XML layout looks as follows:

```
<rowarea name="Test" height="100">
    <rowscrollarea height="100%">
        <itr>
            <label name="Vorname" width="100">
            </label>
            <field valueprop="firstname" width="200">
            </field>
        </itr>
        <itr>
            <label name="Vorname" width="100">
            </label>
            <field valueprop="firstname" width="200">
            </field>
        </itr>
        <itr>
            <label name="Vorname" width="100">
            </label>
            <field valueprop="firstname" width="200">
            </field>
        </itr>
        <itr>
            <label name="Vorname" width="100">
            </label>
            <field valueprop="firstname" width="200">
            </field>
        </itr>
        <itr>
            <label name="Vorname" width="100">
            </label>
            <field valueprop="firstname" width="200">
            </field>
```

```
        </itr>
        <itr>
            <label name="Vorname" width="100">
            </label>
            <field valueprop="firstname" width="200">
            </field>
        </itr>
    </rowscrollarea>
</rowarea>
```

ROWSCROLLAREA Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true"	Optional	true false

	the table itself is sized to fill the maximum height of the available area.		
takefullwidth	If set to "true" then the control takes all available horizontal width as its width. If set to "false" then the control does not have a predefined width but grows with its content.	Optional	true false
areastyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
areaclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much	Optional	true false

	faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.		
vscrollposprop	Name of adapter property that dynamically sets the position of the vertical scrollbar. The value top will scroll up to the top, the value bottom will scroll down to the bottom of the container.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "hidden".	Optional	auto scroll hidden

18 HSPLIT and VSPLIT

- Example for HSPLIT 176
- Example for VSPLIT 177
- HSPLIT Properties 178
- VSPLIT Properties 180
- SPLITCELL Properties 181
- Defining the Split Size 182

HSPLIT or VSPLIT allows to define a container area that is subdivided into two split cells. Between the split cells there is a border. By dragging and dropping the border, you can change the size of the split cells. Each split cell itself is a container that can be used just as normal.

While an HSPLIT control subdivides an area into two split cells by a horizontal line, VSPLIT uses a vertical line.

Example for HSPLIT

The following example shows the usage of the HSPLIT control:



The split area is divided into two cells: a green cell and a red cell. In addition, there is a line at the bottom in which you can provide the split factor.

The XML layout definition is:

```
<rowarea name="HSPLIT Control" height="100%">
    <hsplit height="100%" heighttopprop="heighttop" hsplitstyle="border:1 solid ↩
#000000">
        <splitcell takefullheight="true" cellstyle="background-color: #00FF00">
            <tr height="100%">
                <label name="Top Split Cell" asplaintext="true">
                </label>
            </tr>
        </splitcell>
        <splitcell takefullheight="true" cellstyle="background-color: #FF0000">
            <tr height="100%">
                <label name="Bottom Split Cell" asplaintext="true">
```

```
                </label>
            </tr>
        </splitcell>
    </hsplit>
    <vdist>
    </vdist>
    <itr>
        <hdist width="100%">
        </hdist>
        <label name="Set Top Height" width="100">
        </label>
       <field valueprop="heighttop" width="100" flush="server" validation="[0-9%]+"
              validationuserhint="100, 200, 500, 30%, 50%">
        </field>
    </itr>
</rowarea>
```

You see that the vertical split area consists of

- one VSPLIT definition, and
- two SPLITCELL definitions.

It is not allowed to have more than two split cells inside one HSPLIT container.

The sizing of the split cells can be done by using a property that is referenced by the HSPLIT property `heighttopprop`. The property must return either a percentage value or a pixel value. When the user changes the size by moving the line between the split cells, then the current new pixel width of the left split cell is written back into the property.

Example for VSPLIT

The VSPLIT control is defined in the same way as the HSPLIT control - but now transferred to vertical dimension. It looks like:



The VSPLIT part of the XML layout definition is:

```
<vsplit height="200" widthleftprop="widthleft" vsplitstyle="border: 1 solid #000000">
    <splitcell takefullheight="true" cellstyle="background-color:#00FF00">
        <itr>
            <label name="Left Split Cell" asplaintext="true">
            </label>
        </itr>
    </splitcell>
    <splitcell takefullheight="true" cellstyle="background-color: #FF0000">
        <itr>
            <label name="Right Split Cell" asplaintext="true">
            </label>
        </itr>
    </splitcell>
</vsplit>
```

HSPLIT Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20").	Optional	100 150 200 250 300 250

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		400 50% 100%
heighttop	Definition of the initial height of the top split area. The height either is a pixel value ("100") or a percentage value ("50%"). You can also define the height dynamically by your adapter - see documentation for HEIGHTTOPPROP property.	Optional	1 2 3 int-value
heighttopprop	Name of adapter parameter that specifies the height of the top split area. The value must either be a pixel value ("100") or a percentage value.	Optional	
hsplitstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden

VSPLIT Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
widthleftprop	Name of adapter parameter that specifies the width of the left split area. The value must either be a pixel value ("100") or a percentage value.	Optional	
vsplitstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
overflow	Definition of the vertical scrollbar's appearance.	Optional	auto

	You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".		scroll hidden

SPLITCELL Properties

Basic			
takefullheight	Indicates if the content of the control's area gets the full available height. If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'. Background information: container control's internally open up a table in which you place rows (ITR/TR) which then hold controls (e.g. LABEL/FIELD). The table that is opened up normally has no explicit height and grows with its content as consequence. By specifying "takefullheight=true" the table itself is sized to fill the maximum height of the available area.	Optional	true false
cellstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Defining the Split Size

The split size of HSPLIT and VSPLIT can be set in the following ways:

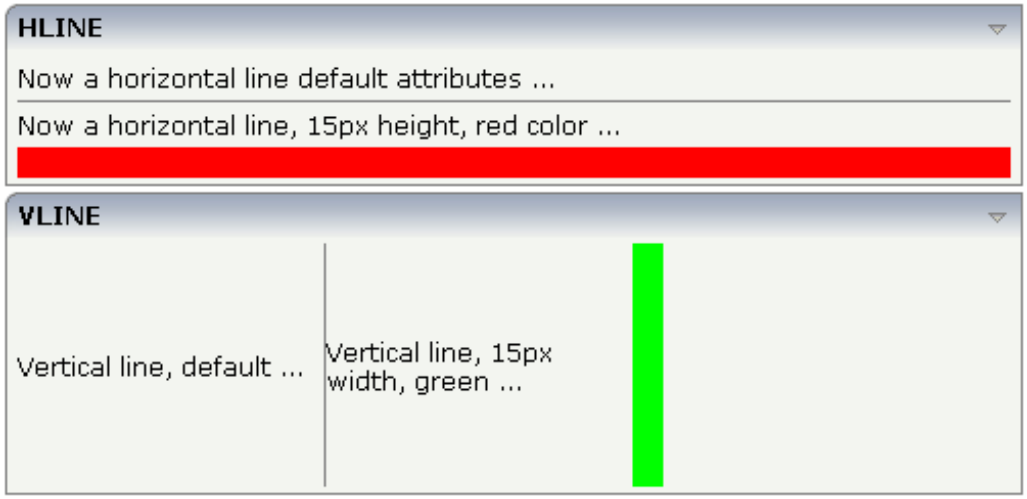
- Fixed definition if initial split size: by using the HSPLIT property `heighttop` and the VSPLIT property `widthleft`, you can preset the size in a “hard way”. The value will be used as the initial size.
- By using the HSPLIT property `heighttopprop` and the VSPLIT property `widthleftprop`, the size can be defined by a server side property. Maybe you have some personalization in which the size is kept for every split area - and proposed the next time the user visits the page.

19 HLINE and VLINE

- VLINE Properties 185
- HLINE Properties 186

Both controls are actually not container controls, but they are typically used for structuring content - this is the reason why they are mentioned here. The controls are rather simple: they represent lines. HLINE represents a horizontal line and VLINE represents a vertical line.

Have a look at this demo:



The corresponding XML layout definition is:

```
<rowarea name="HLINE">
    <itr>
       <label name="Now a horizontal line default attributes ..." asplaintext="true">
        </label>
    </itr>
    <hline>
    </hline>
    <itr>
        <label name="Now a horizontal line, 15px height, red color ..." ↩
asplaintext="true">
        </label>
    </itr>
    <hline height="15" color="#FF0000">
    </hline>
</rowarea>
<rowarea name="VLINE" height="150">
    <itr height="100%">
        <label name="Vertical line, default ..." width="150" asplaintext="true">
        </label>
        <vline>
        </vline>
        <label name="Vertical line, 15px width, green ..." width="150" ↩
asplaintext="true">
        </label>
        <vline width="15" color="#00FF00">
        </vline>
```

```
    </itr>
</rowarea>
```

For each line, you can define its width/height and its color.

VLINE Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
color	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

HLINE Properties

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
color	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

20 Performance Optimization with Containers

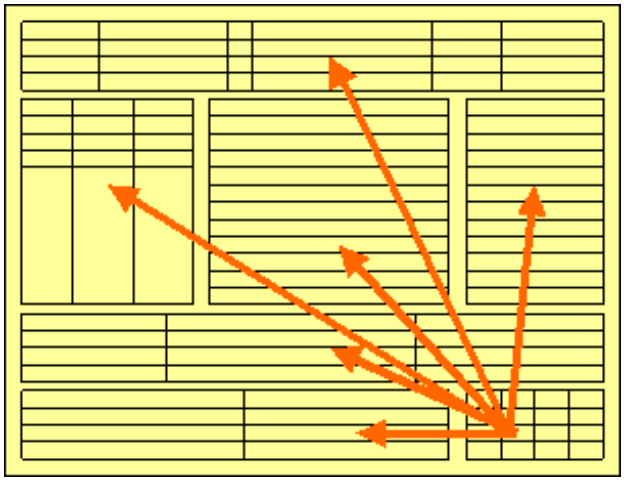
Containers internally use HTML table rendering for arranging their content: inside a container there are rows, inside the rows there are columns and inside the columns there are controls.

HTML table rendering is very powerful: if you have already written pages on your own using an HTML editor, then you know that you can size the container in the following way:

```
<table width='100%'>
<tr>
  <td width='100'>Hallo</td>
  <td width='100%'>Hello world!</td>
  <td><img src='xyz.gif'></td>
</tr>
</table>
```

During rendering time, the browser tries to optimize the table rendering. The browser knows that inside the definitions there is one column that wants to occupy the whole width, one column that wants to have a width of 100 pixels and one column that holds an image. Consequently, it somehow renders the table so that the best result is rendered. This optimization is quite expensive - especially if you have tables nested in tables nested in tables etc.

In nested table scenarios, every little change in one table can have the consequence that the whole HTML table is optimized again.



Since the optimization now happens on several levels, the browser uses a lot of resources to do so. This can be noticed especially if you render pages with a height of 100%: the page is not built by appending one information after the other - but you tell that the controls occupy a certain percentage based height of the whole page.

How can you find that out? If you have got the feeling that a page behaves in a slow way and you are not sure whether it is your server side application or the browser side rendering, then there are two ways to easily find out:

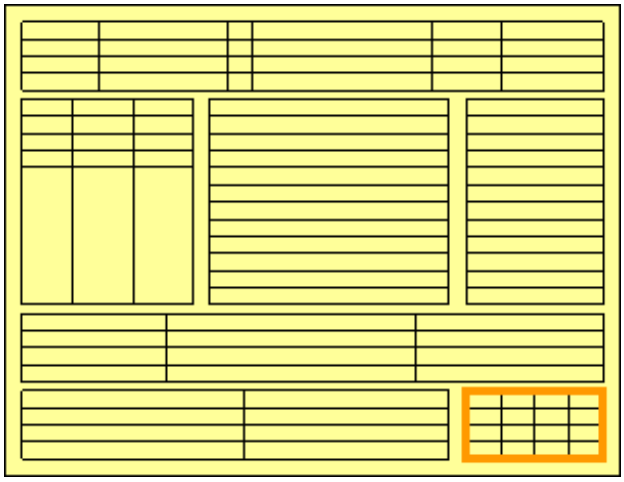
- Look into the Application Designer log file. Each server side request is recorded with its consumption of milliseconds on server side.
- Resize the page in the browser: if this is not fast but takes time, then this is an indicator for bad rendering performance - or in other words: for a lot of optimization that is happening behind the curtain.

But: there are nice ways to speed up the rendering - and to build optimization limits for the browser. Internally, the ways are quite simple, but the consequence can be dramatic.

Most containers support a `fixlayout` property: the possible values are "true" or "false" - "false" being the default. When switching the `fixlayout` property to "true", then the content area of the container is internally arranged in such a way that the area always determines its size from its own width and height specification. The browser does not look into the contents of the area in order to try to optimize the size of the area, but always follows the width and height that you define.

What happens if the controls inside your container area do not fit into the area? What does not fit inside the container area, is cut.

Setting `fixlayout` to "true" means that the browser only optimizes table rendering inside the container - but never outside - because the container has a certain size:



Follow the rules:

- Every time the size of a container area is not determined by its content but is explicitly set by you, switch the `fixlayout` flag to "true".
- The flag only has consequences if you define the width and height of the corresponding container. In cases in which the width is defined by the control (for example, ROWAREA always has a width of 100%), you have to define the height. The height is either defined by a corresponding `height` property or by a `takefullheight` property.

21 ROWTABSUBPAGES and STRAIGHTTABPAGE

- Adapter Interface 192
- Built-in Events 193
- Session Management 193
- Performance Considerations 194
- ROWTABSUBPAGES Properties 194
- STRAIGHTTABPAGE Properties 196

The ROWTABSUBPAGES control allows you to switch between several Application Designer pages such as NATPAGE using tabs. The displayed number of tabs and names are dynamically set by the Natural application at runtime. A page layout may only contain a single ROWTABSUBPAGES control. If you want to render more than one ROWTABSUBPAGES control in a page layout, you have to create a modular structure using **SUBCISPAGE2** controls.

Optionally, the ROWTABSUBPAGES control may contain exactly one STRAIGHTTABPAGE control as a subnode. STRAIGHTTABPAGE must be the first tab. This allows for combining **ROWTABAREA** behavior with ROWTABSUBPAGES behavior. Having a STRAIGHTTABPAGE as the first tab improves the loading behavior of ROWTABSUBPAGES. For an example, see the Natural for Ajax demos.

Adapter Interface

```
1 MYTABPAGES
2 SELECTEDINDEX (A) DYNAMIC
2 TABITEMS (1:*)
3 NAME (A) DYNAMIC
3 PAGEID (A) DYNAMIC
3 PAGENAME (A) DYNAMIC
3 PAGEURL (A) DYNAMIC
```

Element	Description
SELECTEDINDEX	Set the active page. The default is "0".
TABITEMS	Array with a corresponding data object for each tab: `'/cisnatural/NatLogon.html&xciParameters.natsession=Local&xciParameters.natparamext=stack%3D%28LOGON+SYSEXNJX%3BCTRSBI-P%29'`
NAME	The visible text that is to be shown on the tab.
PAGEID	For each PAGEID in the same Application Designer subsession, a new Natural session is created. Identical PAGEID elements within a subsession will refer to the same Natural session. If you do not want to create a new Natural session for this tab, leave PAGEID empty and specify PAGENAME instead. See also ***Session Management*** below.
PAGENAME	PAGENAME is only required if you do not want to create a new Natural session for this tab. In this case, you must leave PAGEID empty. Instead, you specify the name of the page layout in PAGENAME. This can either be the name of the layout without the extension (such as "mylayout") or a relative path (such as "/myuicomponent/mylayout.html"). See also ***Session Management*** below.

Element	Description
PAGEURL	The URL for loading the page. Example: '../servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html& xciParameters.natsession=Workplace& xciParameters.natparam=stack%3D%28LOGON+SYSEXNJX%3BHELLOW-P%29'

Built-in Events

In case the tabs share the same Natural session (see ***Session Management*** below), an event is triggered in the Natural application as soon as a different tab is selected. With this event, the Natural application can apply the data that is required to display the page on the tab. For details, see the corresponding sample in the Natural for Ajax demos.

The name of the triggered event is `reactOnSwitchTo`*`pagename`*, where *`pagename`* is the name of the layout (without the extension) which will be activated with the next event. For example, the event `reactOnSwitchToMylayout` will activate the layout named "mylayout.xml". Note the CamelCase notation for the event name. Even though the first character of the layout name is a lower-case "m", this is always an upper-case "M" within the event name.

Session Management

With the ROWTABSUBPAGES control, you can either use the same Natural session for multiple tabs or you can use a different Natural session for each tab.

If a new Natural session is to be created for a tab, you have to specify a `PAGEID` value for this tab. In this case, the session management for the ROWTABSUBPAGES control is the same as described for the **SUBCISPAGE2** control. See ***Session Management***.

Alternatively, multiple tabs of the ROWTABSUBPAGES control can use the same Natural session. If you do not specify a `PAGEID` value, but specify a `PAGENAME` value for multiple tabs, all NATPAGE pages in these tabs will run in the same Natural session. Note that the Natural session used for the tabs is not the same Natural session as the Natural session used for the NATPAGE which contains the ROWTABSUBPAGES control. In this case, you have two different Natural sessions: one for the NATPAGE containing the ROWTABSUBPAGES control, and another for the tabs in the ROWTABSUBPAGES control.

Performance Considerations

The performance considerations regarding when to use ROWTABAREA and when to use ROWTABSUBPAGES also apply to NATPAGE controls. See the description of the ROWTABSUBPAGES control.

With Natural applications, an additional aspect has to be considered. The ROWTABSUBPAGES control allows you to embed Natural applications which run on different servers. Regarding resources, the Natural application needs to decide which NATPAGE in a "tab" is to use its own Natural session (see also ***Session Management*** above).

ROWTABSUBPAGES Properties

Basic			
pagesprop	Name of the adapter parameter that provides the content of the control	Obligatory	
triggerserver	Flag indicating whether the adapter should be triggered if the user switches between pages. If set to true, method trigger() inside the TABSUBPAGESInfo object is called - before switching the page. Therefore the adapter can abort a page switch - maybe a user has to enter some data first on the current page before switching to another one.	Optional	true false
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
scrollable	If set to "true" then small icons will appear on the right border of the control. If the size of the "tabs" is too big and some tabs are cut	Optional	true false

	as consequence then you can use these icons for scrolling left and right.		
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
fastbufferswitch	If this property is switched to "true" (default is "false") then the contained subpages are buffered in a way that switching between tabs is not done by loading a new page but by just switching the visibility of pages. Please pay attention to that switching between pages in this case does not reload the page content from the server when switching! In order to enable fast switching you have to set the framebuffersize in cisconfig (n +1), n being the number of tabs to switch.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
borderwidth	Border width (in pixels) of the sub-page that is contained inside this control. Define "0" to avoid rendering any border.	Optional	1 2 3 int-value
leftindent	Inserts a horizontal distance left of the first "tab" and shifts the "tabs" to the right as consequence. The value you may define represents the number of pixels that are inserted.	Optional	1 2 3 int-value
paddingleft	Number of pixels which you want to keep as margin between the tab control's left border and the inner sub page. Default is 5 pixel.	Optional	1 2 3 int-value
paddingtop	Number of pixels which you want to keep as margin between the upper tab row and the inner sub page. Default is 5 pixel.	Optional	1 2 3 int-value

paddingright	Number of pixels which you want to keep as margin between the tab control's right border and the inner sub page. Default is 5 pixel.	Optional	1 2 3 int-value
paddingbottom	Number of pixels which you want to keep as margin between the bottom of the tab control and the inner sub page. Default is 5 pixel.	Optional	1 2 3 int-value
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

STRAIGHTTABPAGE Properties

Basic			
id	Id of the TABPAGE. Each page has an id that refers to the PAGE1 .. PAGE9 definition inside the ROW/COLTABAREA control that contains the TABPAGE. Clicking a "tab" will display the TABPAGE with the associated id.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

III Working with Controls

Controls are the elements that are placed inside containers. This part first gives some common rules that are valid for all controls, then describes the controls in more detail.

The information provided in this part is organized under the following headings:

Some Common Rules for all Controls
BREADCRUMB
BUTTON
BUTTONLIST
CHECKBOX
COMBODYN2
COMBOFIX
DATEINPUT
DATEINPUT2
DROPICON
FIELD
FILEUPLOAD/FILEUPLOAD2
ICON
ICONLIST
IHTML
IMAGEOUT
IMAGEVIEWER
LABEL
MENUBUTTON
METHODLINK
MULTISELECT
RADIOBUTTON
SCHEDULELINE

SLIDER
STRIPSEL
SUBCISPAGE2
SUBPAGE
TABSEL
TABSTRIP2
TAGCLOUD
TEXT
TEXTOUT
TOGGLE

Special Controls:

CHART (Deprecated)
RCHART
RPIECHART
OPENSTREETMAP
GOOGLEMAP
REPORT
REPORT2
TIMER

Natural for Ajax Controls:

NJX:BUTTONITEM
NJX:BUTTONITEMFIX
NJX:BUTTONITEMLIST
NJX:BUTTONITEMLISTFIX
NJX:DOCUMENTLINK
NJX:EVENTDATA
NJX:FIELDITEM
NJX:FIELDLIST
NJX:FIELDVALUE
NJX:MASHZONE and BMOBILE:MASHZONE
NJX:NJXFILEDOWNLOAD
NJX:NJXFILEUPLOAD2
NJX:NJXVARIABLE

22 Some Common Rules for all Controls

- Name and Text ID 200
- Table, Row, Column, Control 200
- Explicit Alignment 200
- Binding to Adapter Parameters 201
- Directly Influencing the Control Style 201
- Dynamically Controlling the Visibility and the Display Status of Controls 202
- Focus Management 204
- Flushing of Inputs 205
- Tab Sequence 205
- Tooltips 207
- Images 208
- Documents 209

Name and Text ID

Every time a control needs a static text definition (the name of a button or the name of a label), there are always two possibilities to define this text:

- Specify a name directly.
- Specify a text ID. This is a literal replaced with a string that is determined inside the multi language management at runtime.

Table, Row, Column, Control

Most controls that allow dynamic sizing offer the following properties:

- `colspan` - number of columns occupied by the control.
- `rowspan` - number of rows occupied by the control.
- `width` - width.
- `height` - height.

These properties influence the way how controls are placed into container rows.

Explicit Alignment

Controls are put into table columns. If the column is wider or higher than the control itself, then you can explicitly control the vertical and horizontal alignment of the control inside the columns.

Most controls offer two properties:

- `valign`
  Specifies the vertical alignment. Valid values are "top", "middle", "bottom". "middle" is the default value.
- `align`
  Specifies the horizontal alignment. Valid values are "left", "center", "right". The default value depends on the control. For example, labels are aligned "left" by default, the default for radio buttons is "center".

Pay attention: `valign` and `align` only affect the position of the control inside the column in which it is positioned if the column is larger than the control. If the column is exactly as wide and high as the control itself, which is the typical case, then they do not have any visual effects - and also need not be defined.

`align/valign` do not affect the control's internal alignment.

Binding to Adapter Parameters

Most controls provide properties to specify the binding to the adapter processing. There is a naming convention, which is:

- The names of the properties which specify the binding to an adapter parameter end with "prop".
- The names of the properties which specify the binding to an event end with "method".

The type of the adapter parameter which is referenced by a control depends on the control itself:

- Most controls directly bind to scalar adapter parameters.
- More complex controls bind to an array of group structures.

The type of adapter parameter is described with each control.

Directly Influencing the Control Style

All controls that incorporate textual information - such as labels, buttons or fields - offer the possibility to influence directly the style that is used for displaying the information.

The normal style is derived from the definition inside a cascading style definition file (file *layout.css* inside the *html/general* directory of the server). Overwrite or enhance this style information for your controls by passing the style information inside the corresponding style properties.

The properties specifying the style information end with the suffix "style", e.g. there is a property `labelstyle` for the label tag. The value of the property can be any kind of a valid HTML style specification. If you want to change the display style of a label to be large and blue, define the label in the following way:

```
<label name="Test" width="150" labelstyle="font-size: 24pt; color: #0000FF">
</label>
```

Dynamically Controlling the Visibility and the Display Status of Controls

It is possible to influence the visibility of all input controls (FIELD, BUTTON, etc.) by adapter parameters.

For some of these controls there is a property `visibleprop`, specifying an adapter parameter that returns "true" or "false". By this, you can control whether you want to display the control within the client or not.

Input controls support a property `statusprop` and a property `displayprop`. Using the corresponding adapter parameters, you can dynamically control the display status of the input control. The adapter parameter for the `statusprop` can contain the following values:

INVISIBLE
ERROR
ERROR_NO_FOCUS
FOCUS
FOCUS_NO_SELECT



Note: INVISIBLE is only supported in controls that don't have a `visibleprop` property. Otherwise, use the `visibleprop` instead, provided it is supported.

Responsive controls additionally support the following values:

WARNING
SUCCESS



Note: For SUCCESS a green and for warning an orange border color is used. You can customize the rendering via the css classes `has-warning` and `has-error`. For more information see the *Style Guide* in the *NaturalAjaxDemos*.

The adapter parameter for the `displayprop` specifies whether the control is display-only (TRUE) or whether it can be edited (FALSE). The adapter parameter can contain the values "TRUE" and "FALSE".

The combination of these two parameter values dynamically defines how the controls are rendered at runtime. The following table defines the rendering of the control for the different combinations:

displayprop	statusprop	Control Status
FALSE (default)		EDIT
FALSE (default)	EDIT (deprecated) [1]	EDIT
FALSE (default)	INVISIBLE	INVISIBLE
FALSE (default)	ERROR	ERROR
FALSE (default)	ERROR_NO_FOCUS	ERROR_NO_FOCUS
FALSE (default)	FOCUS	FOCUS
FALSE (default)	FOCUS_NO_SELECT	FOCUS_NO_SELECT
TRUE		DISPLAY
TRUE	DISPLAY (deprecated) [1]	DISPLAY
TRUE	INVISIBLE	INVISIBLE
TRUE	ERROR	ERROR_DISPLAY
TRUE	ERROR_NO_FOCUS	ERROR_DISPLAY
TRUE	FOCUS	DISPLAY
TRUE	FOCUS_NO_SELECT	DISPLAY

[1] For `statusprop`, the above-mentioned deprecated values are still supported to ensure compatibility with older versions. In case you use these deprecated values for `statusprop`, the values for `displayprop` are ignored.

The difference in behavior between "FOCUS" and "FOCUS_NO_SELECT" affects only the FIELD and TEXT controls. For these controls, FOCUS set the focus and selects the complete text inside the control. "FOCUS_NO_SELECT" sets the focus to the control, but does not select the text. For all other controls, "FOCUS_NO_SELECT" behaves like "FOCUS".

For all other controls - and for more complex manipulations of what is visible and not - use the possibility to be able to control the visibility of rows (ITR, TR) or containers (ROWAREA, ROWTABLE0): these controls provide for a visibility parameter and consequently can be switched on and off.

There is an extended management of what the control status "INVISIBLE" means. Most input controls (FIELD, CHECKBOX, etc.) supporting a `statusprop` or a `visibleprop` also support a property `invisiblemode`. The allowed values of `invisiblemode` are:

- **invisible**
  The corresponding control is completely removed. The horizontal space it occupied before is taken out.
- **cleared**
  The corrresponding control is not visible but still occupies its horizontal space.

- **disabled**
  The corresponding control is displayed with a disabled state. This state is only allowed with a certain number of controls (e.g. button and icon).

Focus Management

Sometimes you want to control the keyboard focus inside a page. Here are the internal rules how a page finds out where to put the focus on.

The default reaction is - if a page is displayed for the first time - to put the focus on the first input control (FIELD, CHECKBOX, RADIOBUTTON, etc.) that is available inside a page. After that, you can navigate through the input controls - and the focus is kept stable when interacting with the server.

With `statusprop` - as mentioned in the previous section - you can interrupt this default reaction; there are two possibilities:

- If an input control is set to status "ERROR", it requests the focus automatically. The purpose is to guide the user automatically to those fields that are not correctly entered.
- If an input control is set to status "FOCUS", it is editable - just as normal - and also requests the focus.

If several input controls are requesting the focus at the same time, the focus is put on the first corresponding input control.

Sometimes you want to change the focus management behavior of the framework for specific server round trips. For TABPAGE controls, you sometimes want the framework not only to set the focus to the first focus-requesting input control but also to open the corresponding tab. For some events, you sometimes do not want the framework to automatically set the focus to the first focus-requesting control. The `focusmgtprop` property of the NATPAGE control allows you to control the focus management for single server round trips. Depending on the executed event, the application can define different focus management modes for corresponding server round trips. For more information, see the description of the `focusmgtprop` property of the **NATPAGE** control.

Flushing of Inputs

Most input controls (FIELD, CHECKBOX, RADIOBUTTON, COMBOFIX, etc.) support a property named `flush`. This property controls whether data input from a user causes an immediate synchronisation with the server or whether data input from a user is stored internally within the client and is synchronized with the next flushing event (e.g. when choosing a button).

There are three different values that can be specified with the `flush` property:

- **""(blank)**
  The data is not synchronized after leaving the control. This is the default.
- **server**
  The data is synchronized with the server immediately when the data has been entered, i.e. when the user has left the corresponding input field.
- **screen**
  The data is synchronized within the controls of the screen. This means - if you have two fields displaying the same property - you can synchronize the fields immediately, without interacting with the server.



Tip: On the one hand, it is useful to flush information in a very fine granular way; you can react on wrong entered data immediately - on the other hand, you have to remember that each flush causes network traffic. The screen's data is sent to the server side processing and the screen waits for the response of the server. During this time, the page is blocked for input and the user sees an hour glass popping up in the left top corner of the screen.

Tab Sequence

By default, the tab sequence of the controls of a page is defined by the order of the controls inside the page's XML layout definition. Using the property `tabindex`, this order can be overridden and the order of the tab index can be explicitly defined.

The following example shows a page with three fields and one button with an explicitly defined tab sequence:



The XML layout definition is:

```
<rowarea name="Simple Tab Sequence">
    <itr takefullwidth="true">
        <coltable0 width="50%">
            <itr>
                <label name="First" width="120">
                </label>
                <field valueprop="first" width="120" tabindex="1">
                </field>
            </itr>
            <itr>
                <label name="Third" width="120">
                </label>
                <field valueprop="third" width="120" tabindex="3">
                </field>
            </itr>
        </coltable0>
        <coltable0 width="50%">
            <itr>
                <label name="Second" width="120">
                </label>
                <field valueprop="second" width="120" tabindex="2">
                </field>
            </itr>
            <itr>
                <hdist width="120">
                </hdist>
                <button name="OK" method="onOK" tabindex="4">
                </button>
            </itr>
        </coltable0>
    </itr>
</rowarea>
```

According to the sequence of controls inside the layout definition, the default tab sequence would be: field **First**, field **Third**, field **Second** and button **OK**.

Due to explicitly defining the `tabindex` property for the fields and the button, the tab sequence is now correct: field **First**, field **Second**, field **Third** and button **OK**.

Pay attention:

- Once having started to explicitly set the tab index in a page, you must consequently continue with all controls of the page. Adding new controls without tab index, is internally interpreted as if these controls were defined with tab index "0".
- Equal tab indices in controls are allowed. In this case, the sequence of the controls inside the layout definition defines the tab sequence among the controls with an equal index.
- Moving controls from one location to the other within a page typically means that you have to adapt the tab sequence accordingly.

The tab index usually is a positive integer value. You may define tab index "-1" for excluding certain controls from the tab sequence at all. In this case, the corresponding controls may only be reached by mouse clicking.

Conclusion:

- In typical pages, you do not have to take care of the tab sequence at all because the default (tab sequence by order of controls in page layout) is adequate to the user's experience.
- Only use the explicit definition of the tab sequence if really it is required - the effort for maintaining each tab index with each control should not be underestimated.

Tooltips

Tooltips can be applied to many controls. If the user hovers with the mouse cursor over a control for some seconds, in non-responsive controls, a small yellow box appears showing some more detailed explanation. In responsive controls the tooltips are rendered with more style and stay as long as the mouse cursor is over the control. In responsive controls also icons and arbitrary html can be rendered in tooltips.

The corresponding controls offer the following properties:

- `title`
  Here you can specify a hard-coded text that is used as the tooltip.
- `titletextid`
  Here you specify a text ID that is passed to the multi language management..
- `titleprop`
  Here you can specify a property to dynamically provide the text for the tooltip at runtime. It is mainly supported in responsive controls.
- `titlestraighttext`
  Here you can specify if the text for the tooltip should be rendered as html or as straight text. It is only supported in responsive controls.

Images

This section describes how to apply images that are contained in binary variables on the Natural server to your page layout. The images can be applied statically and dynamically.

Static Images

Many controls provide properties for loading images. In the corresponding image properties, you usually specify an absolute or relative URL such as "../myproject/images/myicon.gif". For images which are contained in binary variables on the Natural server, however, you must use a different URL value such as "nat:myimage1". Example:

```
<itr>
  <icon image="nat:myimage1">
  </icon>
</itr>
```

The prefix "nat:" indicates that the image is contained in a binary variable on the Natural server. "myimage1" is the name that your Natural application uses for this image. There is no need to download or copy the image manually from the Natural server to the application server or web container. This is done automatically by the Natural for Ajax framework.

You simply add an NJX:OBJECTS control to your page layout and provide the image data in the corresponding Natural data fields as shown below:

```
<natpage>
    <njx:objects>
    </njx:objects>
...
    <pagebody>
...
      <itr>
        <icon image="nat:myimage1">
        </icon>
      </itr>
```

For information on how to load the image into the data structure of the NJX:OBJECTS control, see the description of the **NJX:OBJECTS** control.

Dynamic Images

Dynamic images are specified in the same way as static images. You also add an NJX:OBJECTS control to your page layout. This time, you specify the images that are contained in a binary variable on the Natural server in the corresponding dynamic image properties of the control as shown below:

```
<natpage>
    <njx:objects>
    </njx:objects>
...
    <pagebody>
...
      <itr>
        <imageout valueprop="myimageprop">
        </imageout>
      </itr>
```

This allows your Natural program to apply images which are contained in a binary variable on the Natural server and images which reside in the web application layer alternatively to the same dynamic image property at runtime ("myimageprop" in the above example).

For information on how to load the image into the data structure of the NJX:OBJECTS control, see the description of the **NJX:OBJECTS** control.

Documents

This section describes how to embed, download and upload documents that are contained in binary variables on the Natural server to your page layout.

Embedding and Downloading Documents

As with images, you add an NJX:OBJECTS control to your page layout and load the document into the corresponding data structure of the NJX:OBJECTS control. For further information, see the description of the **NJX:OBJECTS** control.

To embed a document into your page layout, you add a **SUBPAGE** control to your page. In the `valueprop` property of the SUBPAGE control, you have to specify the corresponding document URL dynamically at runtime. If you want to force downloading instead of opening the document, add the parameter `DOWNLOAD=TRUE` to the URL (for example, "nat:mydoc.pdf?DOWNLOAD=TRUE").

Via the **NJX:DOCUMENTLINK** control, you can integrate hyperlinks to documents. When clicking on a hyperlink, the corresponding document is opened in a pop-up dialog.

As an alternative to the NJX:DOCUMENTLINK control, you can use the predefined event `showDocumentForXXX` in all controls which support a method property. `XXX` is the name of a property you can choose. For `XXX`, a corresponding Natural field will automatically be generated in your Natural adapter. To show a document, when the `showDocumentForXXX` event is triggered, the Natural program must apply a valid document URL to the `XXX` field. This URL can refer to documents transported in the data structure of the NJX:OBJECTS control. It can also be a normal browser URL for a document which is accessible from within the web application.

Via the **NJX:NJXFILEDOWNLOAD** control, you can integrate hyperlinks which are directly executed by the browser. To force downloading the document, add the parameter `DOWNLOAD=TRUE` to the URL (for example, "nat:mydoc.pdf?DOWNLOAD=TRUE").

See also the **naturaldocument** example in the Natural for Ajax demos.

Uploading Documents

Via the **NJX:NJXFILEUPLOAD2** control you can upload documents from the client to the NJX:OBJECTS data structure. You can also upload dynamically generated PDF documents to the NJX:OBJECTS data structure by triggering corresponding built-in events in the **REPORT** control.

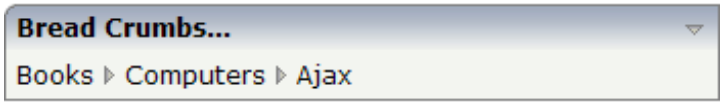
23 BREADCRUMB

- Example 212
- Adapter Interface 212
- Built-in Events 212
- Properties 213

The BREADCRUMB control represents a horizontal list of links. The number of links and the name of each link is dynamically controlled by the application.

The control always occupies 100% of the given width.

Example



The XML layout definition is:

```
<rowarea name="Bread Crumbs...">
    <breadcrumb breadcrumbprop="items">
    </breadcrumb>
</rowarea>
```

Adapter Interface

```
1 ITEMS (1:*)
2 STYLE (U) DYNAMIC
2 TEXT (U) DYNAMIC
2 TOOLTIP (U) DYNAMIC
1 ITEMSINFO
2 SELECTEDITEM (I4)
```

Built-in Events

value-of-breadcrumbprop.onSelect

Properties

Basic			
breadcrumbprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
breadcrumbstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
pixeldistance	Pixel distance between the links that are rendered.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

24 BUTTON

- Example: Simple Button 216
- Example: Button with Image 217
- Hiding and Disabling Buttons 217
- Properties 217

The BUTTON control represents a button. Within the definition, specify an event that is sent to the adapter when choosing the button.

Example: Simple Button



The XML layout definition is:

```
<rowarea name="Buttons">
    <itr>
        <button name="Save As ..." method="saveAs">
        </button>
        <hdist>
        </hdist>
        <button name="Refresh" method="refresh">
        </button>
    </itr>
</rowarea>
```

Example: Button with Image



The XML layout definition is:

```
<rowarea name="Buttons">
    <itr>
       <button name="Save" method="onSave" image="../HTMLBasedGUI/images/save.gif">
        </button>
        <hdist>
        </hdist>
        <button name="Remove" method="onRemove" ↩
image="../HTMLBasedGUI/images/remove.gif">
        </button>
    </itr>
</rowarea>
```

Hiding and Disabling Buttons

Buttons (like many other controls) can be dynamically hidden by using the `visibleprop` property - and referencing to a server side property that decides whether to hide a button or not.

There are two modes of hiding that can be controlled by using the property `invisiblemode`:

- If set to "disabled", the button is grayed and is not selectable anymore.
- If set to "invisible", the button is hidden.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime.	Sometimes obligatory	

	Do not specify a "name" inside the control if specifying a "textid".		
method	Name of the event that is sent to the adapter when the user presses the button. If no method is specified, a default event is sent. If the method starts with javascript: the corresponding javascript method is called.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
name	(already explained above)		
textid	(already explained above)		
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
invisiblemode	This property has three possible values: (1) "invisible": the control is not visible without occupying any space. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more. (3)"cleared": the control is not visible but it still occupies space.	Optional	invisible disabled cleared
switchvisibleproponuserinput	If set to TRUE, the visibleprop is automatically switched to TRUE in case of user input to any input control in this page. The default is FALSE.	Optional	true false
width	Width of the control. There are three possibilities to define the width:	Optional	100 120 140

	(A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
imageheight	Pixel height of image inside button.	Optional	10 20 40 100 300

imagewidth	Pixel width of image inside button.	Optional	10 20 40 100 300
textstyle	CSS style definition that is directly passed into the text of this control. With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are: font-weight: bold color: #FF0000	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants	Optional	VAR1 VAR2 VAR3 VAR4

	"VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!		
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3 4 5

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		50 int-value
imagedisabled	URL of image that is displayed if the control is disabled. Use properties VISIBLEPROP and INVISIBLEMODE to disable the control.	Optional	gif jpg jpeg
submitbutton	Set this property to true and the button will work as an 'Submitbutton', that is necessary if you want to transfer and/or save form values. i.e. password and username or complete search forms Default value is false. You should only use a 'Submitbutton' if the withformtag option of the pagebody tag is set true.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
method	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
nameprop	Name of an adapter parameter that provides the text to be displayed inside the button. Typically buttons have static texts either defined by the property "name" or "textid". Via "nameprop" you can dynamically set the button's text by your application. Use the	Optional	

	nameprop in cases the button's text should change dependent on your logic. Example: you may want to define the button's text to reflect the next status the user can set to a business object.		
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
imageprop	Name of adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	
imagedisabledprop	Name of the adapter parameter that provides as value the URL of the image that is shown when the control is disabled.	Optional	
focusedprop	Name of the adapter parameter which indicates if the control should receive focus.	Optional	
Online help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

25 BUTTONLIST

- Adapter Interface 226
- Properties 226

The button list represents a vertical arrangement of buttons. The number of buttons and the name on each button are dynamically controlled by the application.

The controls always occupy 100% of the given width and occupy the height required by the buttons.

Adapter Interface

```
1 BUTTONLIST (1:*)
2 ID (U) DYNAMIC
2 IMAGEURL (U) DYNAMIC
2 METHOD (U) DYNAMIC
2 STYLE (U) DYNAMIC
2 TEXT (U) DYNAMIC
```

Properties

Basic			
buttonlistprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
pixeldistance	Pixel distance between the buttons that are rendered.	Optional	1 2 3 int-value
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

imageheight	Pixel height of image inside button.	Optional	10 20 40 100 300
imagewidth	Pixel width of image inside button.	Optional	10 20 40 100 300
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

26 CHECKBOX

- Properties 230

The CHECKBOX control displays a check box. It represents a boolean value in the application.

Properties

Basic			
valueprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control.	Optional	left center right

	If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1

			2 5 10 32767
Label			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
hdistpixelwidth	Width of the distance between checkbox and label in pixel.	Optional	
labelstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
valueprop	(already explained above)		
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally	Optional	

	the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.		
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1)	Optional	

	is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

Typically, the CHECKBOX is followed by a LABEL control naming the displayed check box. In the LABEL definition, set the property `asplaintext` to "true".

27 COMBODYN2

- Adapter Interface 236
- Properties 236

The COMBODYN control is the dynamic counterpart of the COMBOFIX control. Whereas the selection options inside the COMBOFIX control are defined in a fixed way inside the page definition, the COMBODYN2 control offers the possibility to control the selection options dynamically in the application.

Adapter Interface

```
1 COSTCENTER (U) DYNAMIC
1 VALIDCOSTCENTERS (1:*)
2 ID (U) DYNAMIC
2 NAME (U) DYNAMIC
2 SELECTED (L)
```

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
validvaluesprop	Name of the adapter parameter that provides the valid values that are available as selectable options.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		

size	Number of rows that are displayed inside the control. If specified as "1" (default) then the control is rendered as combo box - if ">1" then the control is rendered as multi line selection.	Optional	
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	
direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3 4

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		5 50 int-value
renderasfield	If set to "true" then the combo box is rendered like a FIELD control that offers valid value support. Default is "false". The normal translation of COMBODYN2 into HTML renders an HTML-select control. This control has certain limitations inside Internet Explorer: it only offers a very reduced set of styles to manipulate its look and feel and - much worse: it always occupies z-index "0" i.e. if you other areas overlapping the COMBODYN2 area then COMBODYN2 is always on the top. This is quite ugly if e.g. a menu is opened and parts of the menu overlap a COMBODYN2 control.	Optional	true false
allowmultiselection	If set to true then multiple selections are allowed.	Optional	true false
combostyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0

			1 2 5 10 32767
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	xs:string
Binding			
valueprop	(already explained above)		
validvaluesprop	(already explained above)		
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour.	Optional	screen server

	Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.		
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titleprop	(already explained above)		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	

njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

28 COMBOFIX

- COMBOFIX Properties 244
- COMBOOPTION Properties 248

The COMBOFIX control is a selection control. Depending on its configuration, it is either displayed as a combo box or as a selection list.

The COMBOFIX control allows specifying a defined set of values which can be selected. This set of values is defined as part of the layout definition - it cannot be controlled dynamically by the application.



Note: If you want to use dynamic selection, there are two possibilities. Either use the COMBODYN control which has the same look and feel as the COMBOFIX control, but where the selectable values are not specified as part of the page definition and are controlled by the application. Or use the value help pop-up dialogs.

COMBOFIX Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
size	Number of rows that are displayed inside the control. If specified as "1" (default) then the control is rendered as combo box - if ">1" then the control is rendered as multi line selection.	Optional	
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	

direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
combostyle	CSS style definition that is directly passed into this control.	Optional	

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	xs:string
Binding			
valueprop	(already explained above)		
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	

statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

COMBOOPTION Properties

Basic			
name	Name that is displayed as selectable option. Either use the NAME property to specify the text in a "hard" way or use the TEXTID property to define the text in a language dependent way.	Optional	
textid	Text ID that is used for this option. The text id is passed to the multi language management in order to find a language dependent text.	Optional	
value	Actual value of the option that is passed into the adapter property specified by VALUEPROP inside the COMBOFIX control.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

29 DATEINPUT

- Example 250
- Properties 250

The DATEINPUT control is used to input a date or a date with time. The input can be done both with the keyboard or by opening a pop-up in which the user can browse through a calendar. The calendar can be controlled by server side processing in the following way:

- You can define a valid-from and a valid-to date. Thus, the control will not allow the user to input an invalid date.
- You can explicitly control the color and the tooltip information inside the calendar. For example, you may set up a calendar in which vacation times are highlighted in a certain way.

Example

The most simple usage scenario is to just use the DATEINPUT control in the following way:

```
<rowarea name="Dateinput">
    <itr>
        <label name="Order Date" width="120">
        </label>
        <dateinput valueprop="orderDate" width="120">
        </dateinput>
    </itr>
</rowarea>
```

The corresponding screen looks like this:



Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content.	Optional	100 120 140 160 180

	(B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		200 50% 100%
datatype	By default, the DATEINPUT control is managing a day. By explicitly setting a datatype you can define that the control is managing a day and time. In the first use type CDATE within your adapter program - in the second case use type CTIMESTAMP.	Optional	date datetime
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	(already explained above)		
fromprop	Name of the adapter parameter that provides a lower limit for the value of the control. The value is used for client side validation of user input.	Optional	
toprop	Name of the adapter parameter that provides an upper limit for the value of the control. The value is used for client side validation of user input.	Optional	
infoprop	Name of the adapter parameter that provides style information that is used inside the date popup.	Optional	
secondsvisprop	Name of the adapter parameter that provides a boolean that indicates if to show additional seconds. This property make sense only if property DATATYPE is set to "daytime".	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
flush	Flushing behaviour of the input control.	Optional	screen

	By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.		server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Appearance			
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the	Optional	left center right

	align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
inputstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control.	Optional	1 2

	If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		3 4 5 50 int-value
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Valuehelp			
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg

popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cursor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	

njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

30 DATEINPUT2

- Customizing Date and Calendar Formats 258
- DATEINPUT2 – Custom Dates 258
- Properties 259

The DATEINPUT2 control supports the selection and input of a date from a calendar. It has a more modern look and feel than the DATEINPUT control and does not require browser pop-ups.

Customizing Date and Calendar Formats

Default behavior for the date and calendar formats for NATPAGE layouts:

- **Date format in browser:**
  Per default the dates are shown in the browser according to the Natural DTFORM parameter.
- **Date format in Natural program:**
  Depending on the `datatype` property the Natural type D or an (A/U) DYNAMIC type is used. For the latter the format on the server is YYYYMMDD.
- **Calendar format:**
  The first day in week of the calendar is per default Sunday and can be customized per application and per single page at runtime by using the **NJX:SESSIONPARAMS** control

Some use cases require the formats to be customized independently of the central format at design time and in a more flexible way. This is supported by the properties `clientformat`, `severformat`, and `firstdayinweek`. If these properties are set in a control, they will overwrite the default behavior. For the `clientformat` and the `serverformat` property a subset of the Natural date edit masks is supported:

Character	Usage
DD	Day
ZD	Day with zero suppression
MM	Month
ZM	Month with zero suppression
YYYY	Year, 4 digits
YY	Year, 2 digits

DATEINPUT2 – Custom Dates

Per default, the DATEINPUT2 control validates inserted dates according to a range of valid dates. Inserting dates outside of this range are not accepted by the user interface. Sometimes applications use predefined date values to reflect an application-specific semantic meaning. Example: An application may use the date “9999-99-99” to express a never expiring license. For these applications, an end-user must be able to enter these specific custom dates without getting validation errors.

This is supported for DATEINPUT2 controls for which a `clientformat` is set. The custom dates can be specified as the configuration setting `customdates` in the *cisconfig.xml* configuration file.

Example:

```
<cisconfig customdates="99999999;99991231" ….></cisconfig>
```

In the `customdates` property, you can specify several `customdates` separated by a semi-colon. The format for these custom dates is `YYYYMMDD`. When a custom date is entered in the browser no validation error is displayed. They are rendered according to the `clientformat` property value and passed to the server according to the `serverformat` property value if specified.

Example:

```
<dateinput2 clientformat="YYYY-MM-DD" serverformat="DD/MM/YYYY"…>
```

Typing „99999999“ in the browser will be rendered as “9999-99-99” and sent to the server as “99/99/9999”.

For custom dates, the calendar widget is not available. It appears again once the custom date is removed or changed to a “normal date”.

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a	Optional	100 120 140 160 180 200 50% 100%

	width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
datatype	By default, the DATEINPUT control is managing a day. By explicitly setting a datatype you can define that the control is managing a day and time. In the first use type CDATE within your adapter program - in the second case use type CTIMESTAMP.	Optional	date datetime
serverformat	For alphanumeric datatypes you can choose the format of the date at design time. Examples: YYYY-MM-DD, DD.MM.YY, YY/MM/DD. A subset of the Natural date edit masks is supported. The serverforma is the format in which the data is sent to Natural.	Optional	YYYY-MM-DD DD/MM/YYYY MM-DD-YY
clientformat	You can choose the format of the date in the browser at design time. Examples: YYYY-MM-DD, DD.MM.YY, YY/MM/DD. A subset of the Natural date edit masks is supported. If set the Natural DTFORM parameter will not be used. This setting is only for the rendering in the client. It is independent of the set datatype.	Optional	YYYY-MM-DD DD/MM/YYYY MM-DD-YY
firstdayinweek	You can set the first day in week at design time. Valid values are SU - for Sunday - and MO - for Monday. If set the value set by the NJX:SESSIONPARAMS control will not be used.	Optional	SU MO
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	(already explained above)		
fromprop	Name of the adapter parameter that provides a lower limit for the value of the control. The value is used for client side validation of user input.	Optional	
toprop	Name of the adapter parameter that provides an upper limit for the value of the control. The value is used for client side validation of user input.	Optional	
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	

statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
holidaysurlprop	Name of the Adapter paramter which provides the URL for a json file with custom holidays dynamically at runtime.	Optional	
Appearance			

popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cursor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false
popuponF4F7	Per default the calendar is opened on F4 and F7. Set this property to false if you want to use F4 or F7 for other purpose.	Optional	true false
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
numberofmonths	Number of months shown for selection. Default is 1	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR).	Optional	left center right

	Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
inputstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant"	Optional	VAR1 VAR2

	property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!		
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767

holidaysurl	URL for json file, which contains custom holidays.	Optional	
holidaysstyleclass	Name of the css style class, which is used for the rendering of custom holidays. Default is DATEINPUT2Holidays	Optional	
holidaysdescriptionastooltip	Set this property to true if you want to show descriptions in the json file as tool tips.	Optional	true false
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
placeholder	The text for the HTML placeholder attribute. The placeholder attribute specifies a short hint that describes the expected value.	Optional	
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	

njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

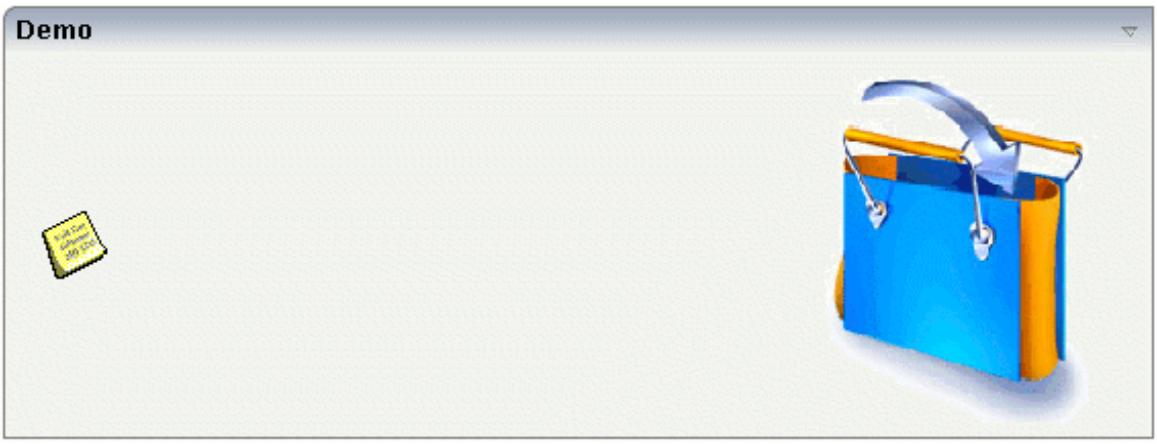
31 DROPICON

- Example 268
- Properties 268

The DROPICON control is an icon that can be used in order to build drag-and-drop scenarios. A DROPICON can be defined as the starting point of a drag-and-drop operation or as the target point of a drag-and-drop operation.

Example

Have a look at the following screen:



The user can click the left mouse button on the left icon (drag), move the mouse to the right icon and then release the mouse button (drop).

The configuration of drag and drop is quite simple: the icon that is used for starting drag-and-drop operations leaves a certain drag information - a plain string. The receiving icon, on which the user performs the drop operation, receives both an event and the string which was left by the icon from where the operation was started.

Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	gif jpg jpeg

draginfo	String containing any kind of application data to identify the source DROPINFO control within a drag and drop process. Use property DROPINFOPROP to return this data on runtime.	Optional	
draginfoprop	Name of the adapter parameter that provides for information that is passed to the adapter when dropping this control over another DROPICON. Do not use this property (or property DROPINFO respectively) if you do not want the user to drag this control.	Optional	
dropinfoprop	Name of the adapter parameter to that the "drag info" of the dragged DROPICON control is set. Do not use this property if this control should not accept other DROPICON controls within a drag and drop process (i.e. is not a drop target).	Optional	
dropmethod	Name of the event that is sent to the adapter when the user is dragging another DROPICON control over this control and drops it there. Do not use this parameter if this control should not accept other DROPICON controls within a drag and drop process (i.e. is not a drop target).	Sometimes obligatory	
method	Name of the event that is sent to the adapter when clicking on the control.	Sometimes obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
draginfoprop	(already explained above)		
dropinfoprop	(already explained above)		
dropmethod	(already explained above)		
imageprop	Name of adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	
method	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
Appearance			
image	(already explained above)		
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible.	Optional	invisible cleared

	(2)"cleared": the control is not visible but it still occupies space.		
imageinactive	If the visibility is dynamically controlled by using the INVISIBLEPROP then there are two ways the icon reacts if the corresponding property passes back "false". If you want the icon to switch into an inactive status then define inside this property the URL of the image that is the inactive counter part to the normal icon image. Maybe the image is a grayed version of the normal icon image. If you do not define a value for this property then the icon is made invisible.	Optional	
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	
withdistance	If set to "true" then 2 pixels of distance are kept on the left and on the right of the icon. Reason behind: if arranging several icons inside one table row (ITR, TR) then a certain distance is kept between the icons when this property is set to "true".	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom

colstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
spanstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Online Help			
title	Text that is shown as tooltip for the control.	Optional	

	Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.		
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		

32 FIELD

- Built-in Events 274
- Properties 274

The FIELD control is used for entering data. It provides the following features:

- Normal input/output of text.
- Password input.
- Autocomplete (see also the **AUTOCOMPLETE** control).
- Dynamic control if input is allowed.
- Dynamic highlighting of field in case of errors.
- Flush the input directly to the server when leaving the field.
- Raise an event on pressing F4 or F7 or on click - useful for value help pop-up dialogs
- Adapt the output to a data type (e.g. transfer "YYYYMMDD" to a visible date field)

Built-in Events

findValidValuesForXXX

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	specify a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
length	Width of FIELD in amount of characters. WIDTH and LENGTH should not be used together. Note that the actual size of the control depends on the font definition if using the LENGTH property.	Optional	5 10 15 20 int-value
maxlength	Maximum number of characters that a user may enter. This property is not depending on the LENGTH property - please do not get confused by the similar naming. MAXLENGTH has nothing to do with the optical sizing of the control but only with the number of characters you may input.	Optional	5 10 15 20 int-value
autotab	If set to true, an automatic tab is executed for fields with a specified MAXLENGTH when the maxlength value is reached. For fields without a MAXLENGTH specified it has no effect. Default is true.	Optional	true false
textalign	Alignment of text inside the control.	Optional	left center right
password	If set to "true", each entered character is displayed as a '*'.	Optional	true false
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr

uppercase	If "true" then all input is automatically transferred to upper case characters.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3 4

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		5 50 int-value
fieldstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
bgcolorprop	Name of the adapter parameter that provides the background color of the control.	Optional	
fgcolorprop	Name of the adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The	Optional	

	color value is used as text color in the control. - The background color is automatically chosen dependent from the text color: for light text colors the background color is black, for dark text colors the color is default. Use BGCOLORPROP to choose both - text and background color.		
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
valueprop	(already explained above)		
alwaysflush	If set to TRUE then a specified server flushmethod is also called in case the value has not changed. The default is FALSE, meaning that a server flushmethod is only called for a changed value.	Optional	true false
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. -	Optional	screen server

	Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.		
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
flushindexprop	Name of a changeindexprop property of another control. On flush='screen" the value of the specified property is automatically increased so that the controls is automatically refreshed. This property is ignored for flush="server"	Optional	
contextmenu	If set to TRUE for a field myfield, method/event reactOnContextMenuMyfield will be called/triggered on right mouse click. In this method/event you can set a contextmenu correspondingly. Please use the attribute CONTEXTMENUMETHOD in case you would like to use a different method/eventname. In case a valid value is specified for the CONTEXTMENUMETHOD attribute, the value for the CONTEXTMENU attribute is ignored. Default value is FALSE.	Optional	true false
contextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in an empty area.	Optional	
onclickmethod	Name of the event that is sent to the adapter when the user clicks.	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user double clicks.	Optional	
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	

statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
valuetextprop	Name of the adapter parameter that provides a "human understandable" description for the value: in some cases you enter an id into a FIELD but want to display the id and a description to the user. At runtime, the values provided by the VALUEPROP- and the VALUETEXTPROP-property are combined into one text (string) that is returned into the FIELD.	Optional	
textidmode	If using property "valuetextprop" then a field knows an id and a text for a certain value. There are three types of display: either both are shown together, separated by an "-" (e.g. "id - text"). Or only text is shown or only the id is shown. If not defined at all then the system's default text id-mode will be chosen. The default mode can be defined as part of the CIS session context.	Optional	0 1 2
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
bgcolorprop	(already explained above)		
fgcolorprop	(already explained above)		
autocallpopupmethod	Name of the adapter parameter that controls that the field's value help event is sent to the adapter with a certain offset (milliseconds) after last key down event.	Optional	true false
lengthprop	Name of the adapter parameter that dynamically provides the width of the FIELD in amount of characters.	Optional	
maxlengthprop	Name of the adapter parameter that provides the maximum number of characters that a user may enter. Consider to use MAXLENGTH to define this number in a static way.	Optional	
Validation			

datatype	By default, the FIELD control is managing its content as string. By explicitly setting a datatype you can define that the control... ...will check the user input if it reflects the datatype. E.g. if the user inputs "abc" into a field with datatype "int" then a corresponding error message will popup when the user leaves the field. ...will format the data coming from the server or coming form the user input: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). In addition value popups are offered for the user automatically for some datatypes: e.g. when specifying datatype "date" the automatically the field provides a calendar input popup. Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	date float int long time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time
editmask	NATPAGE only: A subset of the Natural edit masks is supported depending on the data type.	Optional	
validationrules	Contains information used for Data Validation. Use the Validation Rules Editor to make changes!	Optional	
validation	Regular expression against which the content of the field is checked on client side when the user changes the field. If the validation fails then an error message popup up and informs the user about the wrong input.	Optional	[a-zA-Z0-9_.-] {1,}\\@[a-zA-Z0-9_.-] {1,}\\.\\w{2,}\\d{5} [0-9)(-/+]+
validationprop	Name of the adapter parameter that provides a regular expression for the validation of the field. Works the same way as VALIDATION but in a dynamic way.	Optional	
validationuserhint	If a client side validation fails due to wrong user input then an error popup is opened. If you define a hint inside this property then the hint is output to the user in order to tell in which	Optional	

	way to input the value. The hint is not language dependent.		
validationuserhintprop	If using validation expressions (either property "validation" or "validationprop") then a popup comes up if the user inputs wrong values into a field. Inside this popup a certain text may be added in order to explain to the user what he/she did not correctly input. This text can be either statically defined or dynamically - by using this property.	Optional	
digits	Number that specifiies how many digits are to be displayed (i.e. digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.	Optional	1 2 3 int-value
digitsprop	Name of the adapter parameter that provides information how many digits are to be displayed (i. e. digits before the decimal character). If this feature is used, the DATATYPE property must be set to 'float'.	Optional	
decimaldigits	Specifies the number of displayed decimal digits. If using this feature then the DATATYPE property must be set to 'float'.	Optional	1 2 3 int-value
decimaldigitsprop	Name of the adapter parameter that provides information how many decimal digits are to be displayed (i. e. digits before the decimal character). If this feature is used, the DATATYPE property must be set to 'float'.	Optional	
spinrangemin	An integer value which defines the lower bound of the value range.	Optional	1 2 3 int-value
spinrangemax	An integer value which defines the upper bound of the value range.	Optional	1 2 3

			int-value
Valuehelp			
popupmethod	Name of the event that is sent to the adapter when the user requests value help by pressing F4 or F7 or by clicking into the FIELD with the right mouse button. When using the popupmethod together with the NJX:EVENTDATA control in a grid, then the event name must have the griddataprop name as prefix. Example: mygrid.mypopupmethod. If the POPUPMETHOD is defined, a small icon is shown inside the field to indicate to the user that there is some value help available.	Optional	openIdValueCombo openIdValueHelp openIdValueComboOrPopu
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popupprop	Name of the adapter parameter that provides the information whether a POPUPMETHOD is available or not. This feature is used in scenarios in which a FIELD offers e.g. value help or not, depending on business logic inside the adapter.	Optional	
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cursor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false
popupcombowidth	Pixel width of the standard "openIdValueCombo" popup dialog. Default is field width or at least 150 pixel.	Optional	1 2 3 int-value
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image	Optional	gif jpg

	type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		jpeg
touchpadinput	Boolean property that decides if touch pad support is offered for the FIELD control. The default is "false". If switched to "true" then you can input data into the field via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
onlinehelp			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
placeholder	The text for the HTML placeholder attribute. The placeholder attribute specifies a short hint that describes the expected value.	Optional	
formula	Contains information used by the Formula Editor. Use the Formula Editor to make changes!	Optional	
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example:	Optional	

	ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.		
Add-ons			
autocompleteref	Adds autocomplete functionality to the FIELD control. As value set the id of the AUTOCOMPLETE control.	Optional	
autocompletedropdownicon	Set this property to use your own dropdown icon. URL of an icon, which is used to show the whole list when it is clicked.	Optional	gif jpg jpeg
autocompletedisplayname	Name of the value to be displayed in an additional control.	Optional	
autocompletedisplayref	Sets a reference to an additional control to display additional information on selection. As value set the valueprop of the control in which you would like to display the information.	Optional	
autocompleteresultsref	Sets a reference to an additional control to display the total number of results. Use this when the number of matching items can be very high and you limited the number of displayed items in the dropdown for performance reasons. As value set the valueprop of the control in which you would like to display the total number.	Optional	
autocompletewithdropdown	If set to "TRUE" a dropdown button/icon will be appended to the field. When it is clicked, all items are shown.	Optional	true false
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group	Optional	

	structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
autocallpopupmethodoffset	The offset (milliseconds) after the last key down event for calling the AUTOCALLPOPUPMETHOD. Makes only sense if an AUTOCALLPOPUPMETHOD is specified.	Optional	1 2 3 int-value
formautocomplete	This property only has effects if the withformtag property in the PAGEBODY is activated. In this case you can switch on and off the browser's autocomplete behavior for HTML form tags in single FIELD controls. Default is TRUE.	Optional	on off new-password

33 FILEUPLOAD/FILEUPLOAD2

- FILEUPLOAD 288
- FILEUPLOAD2 289
- FILEUPLOAD Properties 290
- FILEUPLOAD2 Properties 294

The file upload controls simplify the process of uploading files from the client to the server. Two types are available:

- The FILEUPLOAD control is represented by a button. When you choose the button, a dialog appears showing the file upload form (field input and a file selection button).
- With the FILEUPLOAD2 control, you embed the file upload form into your page.

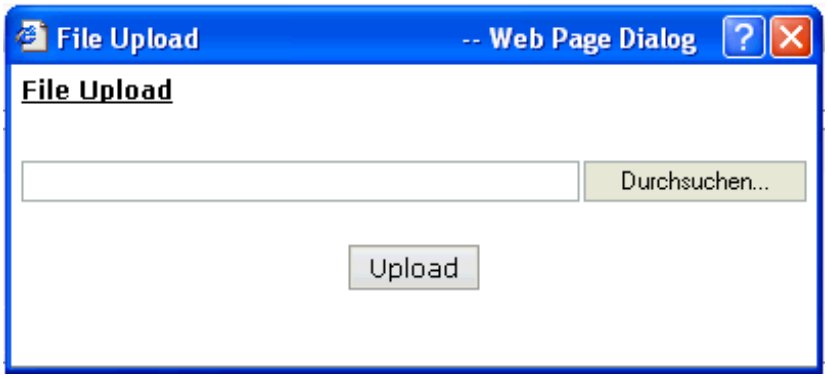
Both types have the program binding, i.e. you can switch between the two types without touching your code.

FILEUPLOAD

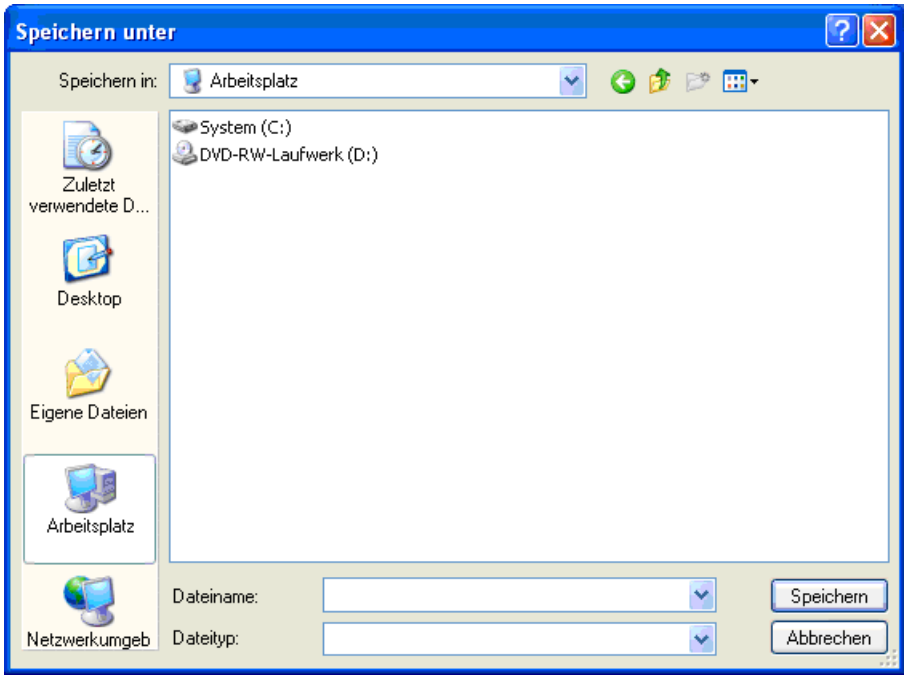
The FILEUPLOAD control simplifies the process of uploading files from the client to the server. Look at the following example:



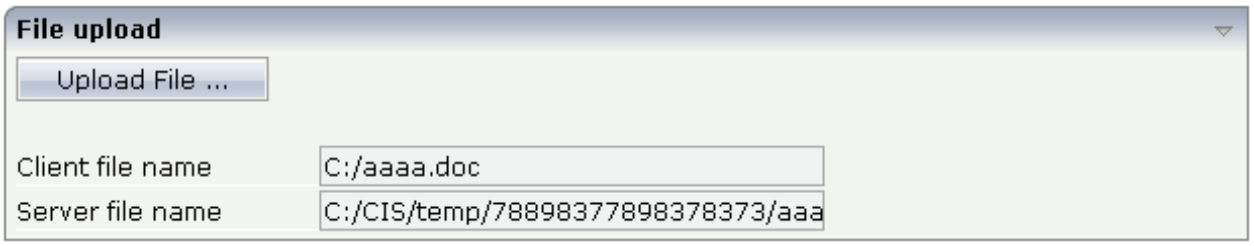
The control - from the look-and-feel perspective - is a button with some special reaction. When you choose the button, the following dialog appears:



You can either enter a file name or you can invoke the file selection dialog by choosing the button to the right of the field (which appears in the language of the browser).



After choosing the **Upload** button, the first screen looks as follows:

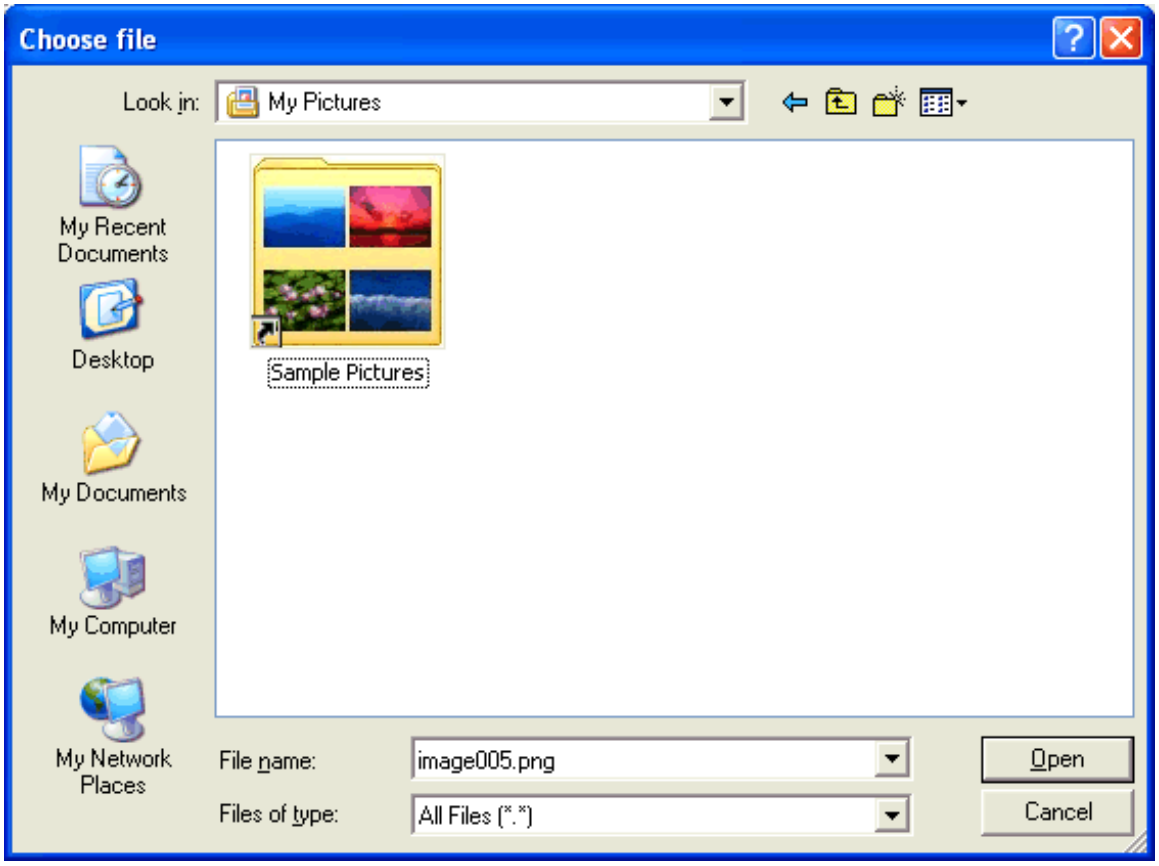


FILEUPLOAD2

With the FILEUPLOAD2 control, you embed the file upload form into your page.



You can either enter a file name or you can invoke the file selection dialog by choosing the button to the right of the field (which appears in the language of the browser).



After choosing the file, the screen looks as follows:



FILEUPLOAD Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
cfileprop	Name of the adapter parameter in which the client file name is passed at upload time.	Obligatory	

sfileprop	Name of the adapter parameter in which at upload time the name of the target file is written, which is a copy of the client file in the server file system. Note that this file name is not the same as the client file name.	Obligatory	
method	Name of the event that is sent to the adapter when a file is uploaded. The file data is available on the server at the point of time this method is called.	Obligatory	
maxfilesize	Maximum size of the file. The unit is specified in the filesizeunit property.	Optional	1 2 3 int-value
filesizeunit	The unit for the maxfilesize property. Default is bytes.	Optional	bytes KB MB GB
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Optional	100 120 140 160 180 200

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
invisiblemode	This property has three possible values: (1) "invisible": the control is not visible without occupying any space. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more. (3)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press	Optional	

	right mouse-button in your browser and select the "View source" or "View frame's source" function.		
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
Binding			
cfileprop	(already explained above)		

sfileprop	(already explained above)		
method	(already explained above)		
visibleprop	(already explained above)		
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

FILEUPLOAD2 Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
cfileprop	Name of the adapter parameter in which the client file name is passed at upload time.	Optional	
sfileprop	Name of the adapter parameter in which at upload time the name of the target file is written, which is a copy of the client file in the server file system. Note that this file name is not the same as the client file name.	Optional	
method	Name of the event that is sent to the adapter when a file is uploaded. The file data is available on the server at the point of time this method is called.	Optional	
maxfilesize	Maximum size of the file. The unit is specified in the filesizeunit property.	Optional	1 2

			3 int-value
filesizeunit	The unit for the maxfilesize property. Default is bytes.	Optional	bytes KB MB GB
withsubmitbutton	If set to "TRUE" adds an additional button to the control to start the file upload.	Optional	true false
submitbuttonname	The name of the submit button in case WITSUBMITBUTTON is set to "true".	Optional	
submitbuttontextid	"Textid" for the name of the submitbutton if WITHSUBMITBUTTON is set to "true". The "textid" is translated into a corresponding string at runtime. Do not specify WITHSUBMITBUTTONNAME if specifying WITHSUBMITBUTTONID.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
cfileprop	(already explained above)		
sfileprop	(already explained above)		
method	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible disabled cleared
Appearance			
invisiblemode	(already explained above)		
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but	Optional	1 2 3

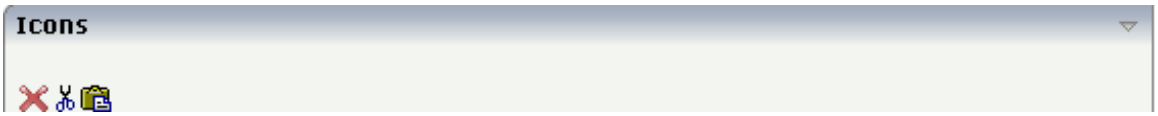
	you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
darkbackground	Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used to integrate light coloured controls into darker container areas.	Optional	true false
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	

34 ICON

- Example 298
- Properties 298

The ICON control is similar to the BUTTON control, but it uses an image to display its function. When chosen, it sends an event to the adapter.

Example



The XML layout definition is:

```
<rowarea name="Icons">
    <itr>
        <icon image="../HTMLBasedGUI/images/remove.gif" method="remove" ↩
title="Remove">
        </icon>
       <icon image="../HTMLBasedGUI/images/cut.gif" method="cut" withdistance="true"
              title="Cut">
        </icon>
       <icon image="../HTMLBasedGUI/images/paste.gif" method="paste" title="Paste">
        </icon>
    </itr>
</rowarea>
```

Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	gif jpg jpeg

imagertl	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Optional	gif jpg jpeg
method	Name of the event that is sent to the adapter when clicking on the control.	Obligatory	
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	10 20 40 100 300
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	10 20 40 100

			300
textsize	The HTML font size of the text. Corresponding to the HTML definition "1" means "smallest" and "6" means "biggest".	Optional	1 2 3 4 5 6
imageinactive	If the visibility is dynamically controlled by using the INVISIBLEPROP then there are two ways the icon reacts if the corresponding property passes back "false". If you want the icon to switch into an inactive status then define inside this property the URL of the image that is the inactive counter part to the normal icon image. Maybe the image is a grayed version of the normal icon image. If you do not define a value for this property then the icon is made invisible.	Optional	gif jpg jpeg
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column.	Optional	top middle

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.		bottom
colstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
spanstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then	Optional	invisible cleared

	there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.		
switchvisibleproponuserinput	If set to TRUE, the visibleprop is automatically switched to TRUE in case of user input to any input control in this page. The default is FALSE.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
nameposition	Position of the (optional) text to the icon. Aside or below, default is aside. Set the corresponding text in the name or the text id property.	Optional	aside below
displaymenuindicator	If set to true a small indicator signals that there is a corresponding menu 'behind this icon'. Default is false.	Optional	true false
Binding			
method	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
imageprop	Name of adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	

imageinactiveprop	Name of the adapter parameter that provides as value the URL of the image that is shown when the control is inactive.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
Deprecated			
withdistance	If set to "true" then 2 pixels of distance are kept on the left and on the right of the icon. Reason behind: if arranging several icons inside one table row (ITR, TR) then a certain distance is kept between the icons when this property is set to "true".	Optional	true false

35 ICONLIST

- Adapter Interface 306
- Built-in Events 306
- Properties 306

The ICONLIST is very similar to the BUTTONLIST, representing a list of items instead of a list of buttons. The list can either be a vertical list or a horizontal list.

Adapter Interface

```
1 ICONLIST (1:*)
2 DRAGINFO (U) DYNAMIC
2 DROPINFO (U) DYNAMIC
2 ID (U) DYNAMIC
2 IMAGEURL (U) DYNAMIC
2 METHOD (U) DYNAMIC
2 NAME (U) DYNAMIC
2 TEXT (U) DYNAMIC
```

Built-in Events

value-of-iconlistprop.onDrop
value-of-iconlistprop.onSelect

Properties

Basic			
iconlistprop	Name of the adapter parameter that represents the control in the application.	Obligatory	
vertical	Direction of the icon list. If not specified (or set to "true") then the icons are arranged in one column, one below the other. If specified as "false" then the icons are arrange in one row, one aside the other.	Optional	true false
cellspacing	An icons of the ICONLIST control is embedded into an internal cell. The CELLSPACING property defined the number of pixels that are kept between the icon an the border of this cell. Use the CELLSPACING in order to define a certain distance each icon keeps from the next item.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Appearance			
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	10 20 40 100 300
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	10 20 40 100 300
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
tablestyle	Style definition (following CSS style sheet definitions) that is used for the background area of the ICONLIST control.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
cellstyle	Style definition (following CSS style sheet definitions) that is used for each cell area of the ICONLIST control in which an icon is kept.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

displaymenuindicator	If set to true a small indicator signals that there is a corresponding menu 'behind this icon'. Default is false.	Optional	true false
additionaltextposition	Position of the text that is displayed inside the control. Use method ICONLISTItem.setName to set the text.	Optional	aside below
textsize	The HTML font size of the text. Corresponding to the HTML definition "1" means "smallest" and "6" means "biggest".	Optional	1 2 3 4 5 6
withrightpadding	Flag (boolean) that indicates whether to insert a padding right hand of the last icon. This attribute does apply for horizontal ICONLIST only (see attribute VERTICAL). Default is true.	Optional	true false
withinvisibleprop	If set to true, a field invisible will be generated into the adapter interface.	Optional	true false
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

36 IHTML

- Properties 310

The IHTML control is used to embed server side generated HTML inside a page that is provided by the application. The IHTML control is very flexible on the one hand. On the other hand, you have to take care about what is defined inside the IHTML area.

Use this control if you have, for example, a server side report generation program already producing HTML as output which you want to include into your pages, etc.

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an	Optional	100 150 200 250 300 250 400 50% 100%

	ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
ihtmlstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
valign	Vertical alignment of control in its column.	Optional	top

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.		middle bottom
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

37 IMAGEOUT

- Properties 314

The IMAGEOUT control is used to present images inside a page. The name of the image is not statically defined inside the layout but is controlled by the application through an adapter parameter.

Properties

Basic			
valueprop	Name of the adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element	Optional	

	does not specify a width then the rendering result may not represent what you expect.		
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

38 IMAGEVIEWER

- Adapter Interface 318
- Example 319
- Properties 319

The IMAGEVIEWER control is used to display images inside a web page. The image can be rotated, zoomed and translated (that is, moved into a specified direction).

The image formats that can be displayed with the IMAGEVIEWER control depend on the capabilities of your web browser. The following image formats normally work in all browsers: JPEG, GIF and PNG.

You can use the mouse wheel to zoom in and out of the page. When an image is zoomed so that its full content does not fit into the page, you can use the mouse to move another portion of the image into view: press and hold the left mouse button inside the image and then move the mouse.

Adapter Interface

```
1 IMAGE
2 IMAGEURL (U) DYNAMIC
2 IMAGEHEIGHT (I4)
2 IMAGEWIDTH (I4)
2 ROTATION (I4)
2 XCENTER (I4)
2 YCENTER (I4)
2 ZOOMFACTOR (F4)
```

Element	Description
IMAGEURL	The path to the image to be displayed.
IMAGEHEIGHT(*)	The height of the image in pixels.
IMAGEWIDTH(*)	The width of the image in pixels.
ROTATION	The rotation parameter in degrees. Valid values: 0, 90, 180 or 270. If using a different value (for example, 45), the values are rounded to the next valid value.
XCENTER(*)	The current center point of the image on the x-axis (that is, the position of the corresponding pixel on the x-axis).
YCENTER(*)	The current center point of the image on the y-axis (that is, the position of the corresponding pixel on the y-axis).
ZOOMFACTOR(*)	The zoom factor of the image inside the image viewer. When setting the zoom factor to 0, the image is zoomed so that it fits completely into the control. A zoom factor of 1.0 shows the image in its original size.

(*) The marked values are not available at program start. They are calculated when the image is loaded into the web page. After the image has been loaded into the web page, the `loadmethod` is triggered. This event arrives as `*PAGE-EVENT` in the server page. To get the information of the marked elements back from the page, you will have to wait for the `loadmethod` event to be triggered by the page. When you use the mouse to move another portion of the image into view, the page is synchronized with the server by sending the `movemethod` information. When you use the mouse

wheel, the page is synchronized with the server by sending the methods which have been defined with the `zoominmethod` and `zoomoutmethod` properties (see ***Properties*** below).

Example

An example which shows the usage of the IMAGEVIEWER control is provided in the Natural for Ajax demos. See the program `CTRIMV-P.NSP`.

Properties

Basic			
valueprop	Name of the adapter parameter that provides as value the URL of the image that is shown inside the control.	Obligatory	
width	Width of the control in pixels.	Optional	10 20 40 100 300
height	Height of the control in pixels.	Optional	10 20 40 100 300
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value

rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
withscrollbar	If set to TRUE scrollbars are shown. Per default no scrollbars are shown.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	(already explained above)		
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
loadmethod	Name of the event that is sent to the adapter when the image is loaded.	Optional	
movemethod	Name of the event that is sent to the adapter when the image is moved in the browser.	Optional	
zoominmethod	Name of the event that is sent to the adapter on zoom in of the image in the browser.	Optional	
zoomoutmethod	Name of the event that is sent to the adapter on zoom out of the image in the browser.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	

njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

39 LABEL

- Example 325
- Aligning the Text 325
- Properties 326

The LABEL control is a static text. The tag has different properties to control the design of the label. It can be used to display plain text or as a headline of a grid.

By default, the label is rendered with a white line under the text. The default is suitable if a FIELD control follows the label.

Example



The XML layout definition is:

```
<rowarea name="Label Controls">
    <itr>
        <label name="Narrow" width="50">
        </label>
        <hdist>
        </hdist>
        <label name="Wide" width="150">
        </label>
        <hdist>
        </hdist>
        <label name="Plain" width="100" asplaintext="true">
        </label>
        <hdist>
        </hdist>
        <label name="Headline" width="100" asheadline="true">
        </label>
    </itr>
    <vdist>
    </vdist>
</rowarea>
```

For a better separation between the LABEL controls, horizontal distances (HDIST) were added.

Aligning the Text

Use the property `textalign` in order to align the label's text. Do not use the `align` property. `textalign` refers to the text inside the control, `align` refers to the position of the control inside the surrounding cell - if the cell is larger than the control.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
propref	If the grid column visualizes data input the name of the property here. This property is located within the row item class. Example: if you use a FIELD or CHECKBOX control input the value of property VALUEPROP here. If the grid column does not visualize any data (e.g. you use a BUTTON control) input an unique column identifier. The PROPREF property is used as key when flushing 'column change events' to the application.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
nowrap	If the textual content of the control exceeds the size of the control then the browser automatically breaks the line and arranges the text accordingly.	Optional	true false

	You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.		
width	(already explained above)		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
asheadline	If set to true, the label has a dark background and the text is written in white (if using the standard style sheet). You may use this rendering style is you use labels as headlines of control grids (ROWTABLEAREA2 control).	Optional	true false
asplaintext	If set to true, no white line is drawn under the label text (if using the standard style sheet). You may use this rendering style if the label is used to name a RADIOBUTTON control or a CHECKBOX control.	Optional	true false
textalign	Horizontal alignment of the text that is shown.	Optional	left center right
cuttext	Boolean property defining the rendering if the text of the label does not fit into the defined width. If "true" then the text is cut - the part that does not fit is hidden. If "false" then the browser opens a second line. Default is "false".	Optional	true false
labelstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:	Optional	background-color: #FF0000 color: #0000FF

	border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		font-weight: bold
labelstyleclass	CSS style class used for rendering.	Optional	
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2 VAR3 VAR4
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By	Optional	1 2 3

	default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

40 MENUBUTTON

- Example 332
- MENUBUTTON Properties 333
- MENUITEM Properties 335

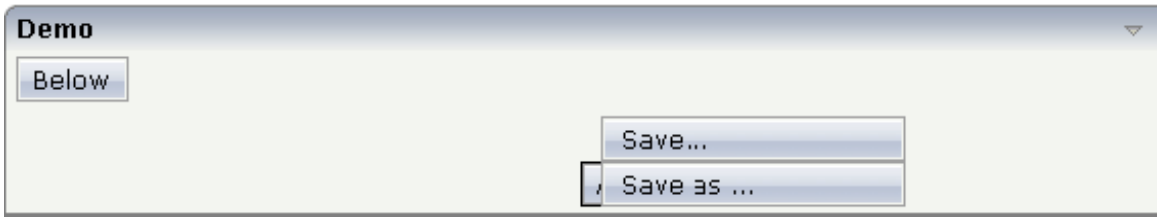
The MENUBUTTON control offers the possibility to arrange buttons in a hierarchy.

Example

In the following example, there are two menu buttons which act differently when they are selected:







The XML code for the example looks as follows:

```
<rowarea name="Demo">
    <itr takefullwidth="true">
        <coltable0 width="50%" takefullheight="true">
            <itr>
                <menubutton name="Below" menuposition="below">
                    <menuitem name="New..." method="newFile" pixelwidth="150">
                    </menuitem>
                    <menuitem name="Open..." method="openFile" pixelwidth="150">
                    </menuitem>
                </menubutton>
            </itr>
        </coltable0>
        <coltable0 width="50%">
            <vdist height="50">
            </vdist>
            <itr>
```

```
                <menubutton name="Above" menuposition="above">
                    <menuitem name="Save..." method="saveFile" pixelwidth="150">
                    </menuitem>
                 <menuitem name="Save as ..." method="saveAsFile" pixelwidth="150">
                    </menuitem>
                </menubutton>
            </itr>
        </coltable0>
    </itr>
</rowarea>
```

In the definition of a menu item, an event that is to be sent to an adapter is exactly defined like with a normal button.

MENUBUTTON Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
menuposition	above if the menu should popup above the base menu button - below if the menu should popup below the base menu button. The default is below.	Optional	above below
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of	Optional	100 120 140 160 180 200 50% 100%

	"100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	

MENUITEM Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
method	Name of the event that is sent to the adapter when clicking on the control.	Obligatory	
pixelwidth	Width of the control in pixels.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
pixelheight	Height of the control in pixels.	Optional	
itemstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	

41 METHODLINK

- Properties 338

The METHODLINK is a control that renders a text that is dynamically provided by the application through an adapter parameter. The text is rendered as a hyperlink. When clicking on the hyperlink, an event is sent to the adapter. It is used in scenarios in which users are in the habit of following links instead of choosing buttons or icons.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
method	Name of the event that is sent to the adapter when clicking on the control.	Obligatory	
valueprop	Name of the adapter parameter that provides the text that is shown as link.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
linkstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
linkclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside	Optional	left center right

	the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
nowrap	If the textual content of the control exceeds the size of the control then the browser automatically breaks the line and arranges the text accordingly. You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.	Optional	true false
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value

Binding			
valueprop	(already explained above)		
method	(already explained above)		
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
linkstatusprop	Name of the adapter parameter that dynamically defines how the link should be rendered and how it should act. Valid values are "DISPLAY" and "EDIT".	Optional	
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in an empty area.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
pseudoclassessupport	Set this attribute to TRUE when you have set an own CSS style class for the LINKCLASS property which uses pseudo classes like ':hover' and ':link'.	Optional	true false

	A HREF attribute will be created, which is required to make the pseudo classes work correctly. Set this property only if you use pseudo classes. Note: If you set this property the text "javascript:void(0)" will be displayed in the status bar when the mouse pointer hovers over the link. This cannot be avoided.		

42 MULTISELECT

- Example 344
- Adapter Interface 344
- Properties 344

The MULTISELECT control allows comfortable input of multiple selections of items from a defined number of items.

Example



The available items are rendered on the left and are brought to the right by choosing the corresponding button. There are buttons to bring all items from the left to the right, and back.

Adapter Interface

```
1 TOWNS (1:*)
2 ID (U) DYNAMIC
2 SELECTED (L)
2 TEXT (U) DYNAMIC
```

Properties

Basic			
valueprop	Name of the adapter parameter representing this control in the application.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Obligatory	100 120 140 160 180

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
withupdown	If set to true, corresponding up and down arrows appear on the right hand side. These arrows allow for changing the order of the selected items.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right

valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
msstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
Binding			
valueprop	(already explained above)		

flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			

testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

43 RADIOBUTTON

- Properties 350

The RADIOBUTTON control displays the radio button. Radio buttons can be grouped together so that a group of RADIOBUTTON controls manipulates one adapter parameter. Each RADIOBUTTON instance represents one value for the adapter parameter.

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
value	Value that represents this instance of the RADIOBUTTON control. The value is set into the adapter property that is defined by the VALUEPROP property when the user clicks onto the control. - Vice versa: the control is switched to "marked" when the adapter property holds the value defined.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
align	Horizontal alignment of control in its column.	Optional	left

	Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible.	Optional	invisible cleared

	(2)"cleared": the control is not visible but it still occupies space.		
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	xs:string
Label			
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Optional	
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
hdistpixelwidth	Width of the distance between checkbox and label in pixel.	Optional	
labelstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
Binding			
valueprop	(already explained above)		
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to	Optional	

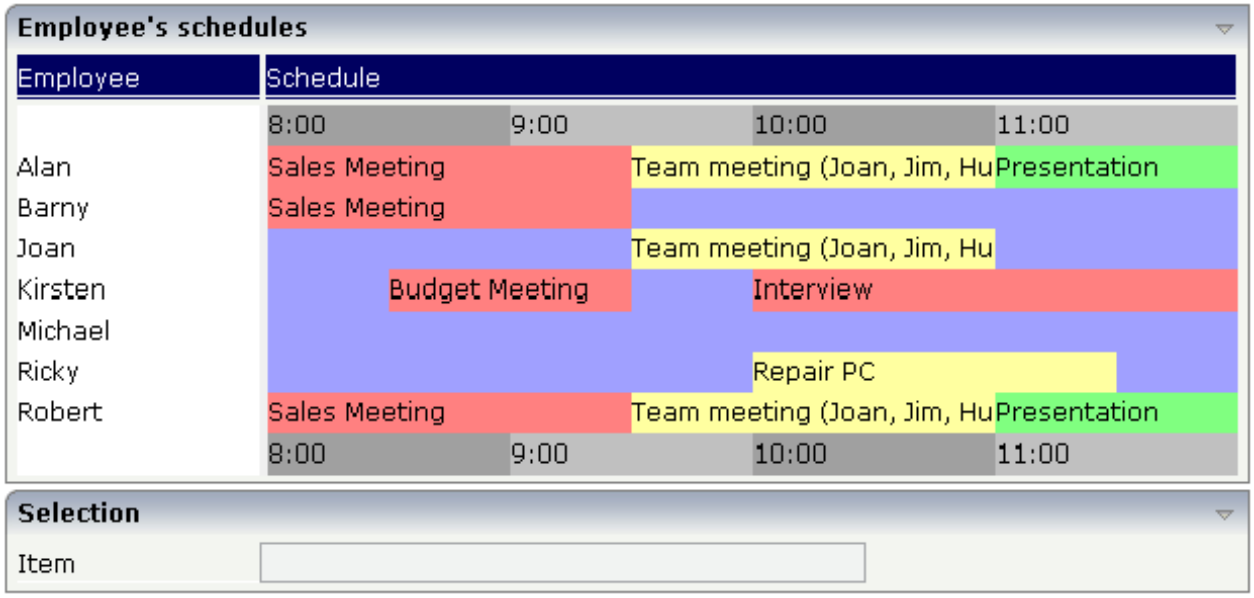
	be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.		
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

The RADIOBUTTON control is typically followed by a label explaining its meaning.

44 SCHEDULELINE

- Properties 356

The SCHEDULELINE control is used to define screens like the following:



You can display a certain sequence of items, each item holding a text, a color value, a size and an identifier. When clicking on an item, a certain event is sent to your adapter and the ID of the selected item is returned to perform activities in your program.

Properties

Basic			
valueprop	Name of the adapter parameter that represents the control in the adapter. It returns a semicolon separated list of schedule items. Each item is represented by a color, a width, a text and a selection id. The width is not a pixel width but represents a "portion" that this schedule item represents. Example: #FF0000\:1000;Text 1;1;#00FF00;500;Text 2;2 The total "logical width" is 1500. The first item occupies 2/3 of the width, the right item occupies 1/3 of the width. The selection is required in case you want to react on user selections. If a user clicks onto one schedule item then the adapter is notified by a certain event - the id of the schedule item is passed as reference. Please have a look into the corresponding property descriptions.	Obligatory	

width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
pixelheight	Height of the control in pixels.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
pixelheight	(already explained above)		
pixelsizemode	A schedule line consists of sections, each one rendered with a certain width. By default the width does not represent a pixel value but represents a logical size. The width of the section depends on the logical size of one section compared with the logical size of the other sections. When switching this property to "true" then the size of the sections are interpreted as real pixel values.	Optional	true false
cellalign	Horizontal alignment of the text inside the control's schedule items.	Optional	left center right
cellvalign	Vertical alignment of the text inside the control's schedule items.	Optional	top middle bottom
cellstyle	Style that is used inside the schedule item cells. Can be any CSS style.	Optional	background-color: #FF0000

			color: #0000FF font-weight: bold
cellnowrap	If switched to "true" then the text inside the schedule item cells is not broken if exceeding the size of the control - the text is cut instead. Default is "false".	Optional	true false
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
crosslineids	Flag (true | false) that indicates that cells of different lines (within ROWTABLEAREA2) does not have same ids. If set to false the control is able to detect and skip unnecessary re-draws (performance).	Optional	true false
tablestyle	CSS style definition that is directly passed into this control.	Optional	background-color: #FF0000

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		color: #0000FF font-weight: bold
Binding			
valueprop	(already explained above)		
selectmethod	Name of the event that is sent to the adapter when the user selects one schedule item with the mouse.	Optional	
selscheduleprop	Name of an adapter parameter in which the id of the selected schedule item is passed.	Optional	
seltypeprop	Name of an adapter parameter that is used in the following way: If the user selects an item it can also be determined, if the item is selected by the left or by the right mouse button. In case the user uses the left mouse button, the value LEFT is passed into the property, which is referenced by the SELTYPEPROP property. In case the user uses the right mouse button, the value RIGHT is passed.	Optional	
preselectmode	If set to "true" then schedule items holding an id can be "preselected": the user can click on a schedule item and it is "grayed" as consequence - without directly calling the selection method. The selection method is called when double clicking onto the schedule item. Default is "false". The reaction of the control when clicking with the right mouse button remains untouched: still the selection method is called by a single right mouse button click.	Optional	true false
Vertical			
verticalschedule	Flag that indicates if the line is rendered vertically. Default is false.	Optional	true false

tooltipprop	Name of an adapter parameter that contains the comma separated list of help texts that are displayed on mouse over (tooltip).	Optional	
imageprop	Name of an adapter parameter that returns a comma separated string of image URLs. An URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/green.gif;;red.gif"	Optional	
imageorientation	Flag that indicates to render the image at the left or right hand of the text.	Optional	left right
dropinfoprop	Name of the adapter parameter to that the id of the dragged cell is set. Do not use this property if you do not want to support drag and drop within the SCHEDULELINE. The server side property needs to be of type "String".	Optional	
onmovemethod	Name of the event that is sent to the adapter on drop of one cell (source) over another cell (target). Use property DROPINFOPROP to get the id of the dragged cell (source). Use SELSCHEDULEPROP to get the id of the cell that got the drop (target).	Optional	
controlkeyprop	Name of an adapter parameter to that the information is set whether the user pressed the CTRL key when selecting a cell.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance,	Optional	

	uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.		

45 SLIDER

- Example 364
- Adapter Interface 365
- Properties 365

The SLIDER control represents a slider. The main use of the slider is to limit the user input to specific values. It uses a number representation for its values, but the numbers can also be used to express string values.

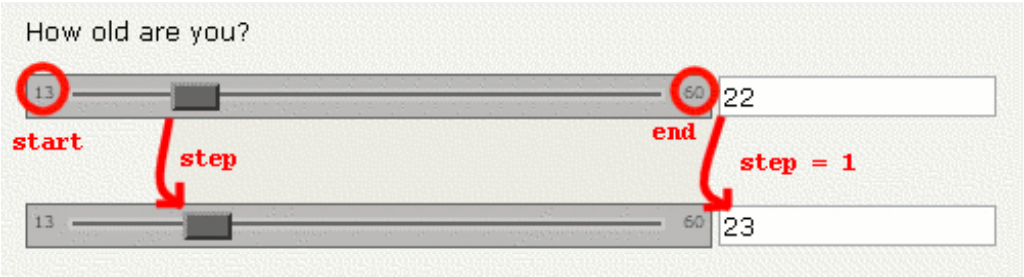
Example



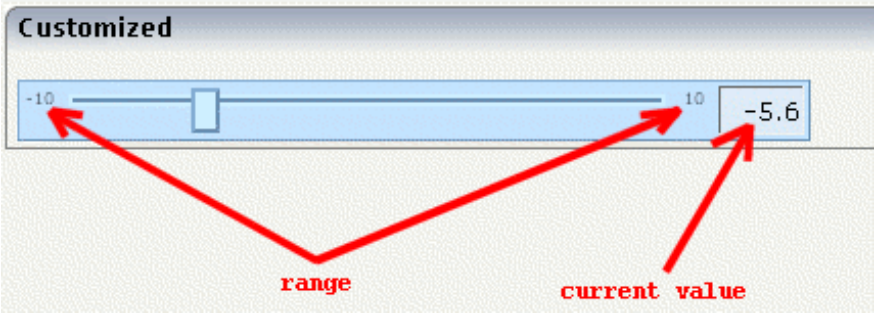
The XML layout definition is:

```
<rowarea name="Number Output">
    <itr>
        <slider valueprop="slider1" from="13" to="60" showrange="true"
                                                showcurrentvalue="false">
        </slider>
    </itr>
</rowarea>
```

The control can be customized by setting its start value, end value and a step. The start and end values form a closed interval. The step defines the distance between two valid values represented by the slider in this interval.



In the above example, the value for the step is the default value "1". The possible values represented by the slider are the integers from "13" to "60". It is possible to specify a floating-point number as a step, for example "0,25". The slider can be further customized by setting the properties `showrange` and `showcurrentvalue` which show the range (start and end value) and the current value of the slider while the user is moving it. The width and height of the slider point is adjustable. The slider point is the element which the user drags and drops. The colors, the borders of the slider, the point, the line, the range and the current value can also be customized.



Adapter Interface

```
1 SLIDER
2 DISPLAYONLY (L)
2 FROM (F4)
2 SLIDERVALUE (F4)
2 STEP (F4)
2 TO (F4)
```

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
Appearance			
width	Width of the slider. Can be given in pixels or percentage.	Optional	100 120 140 160 180 200 50% 100%
displayonly	If set to true, the SLIDER will not be accessible for input. It is just used as an output.	Optional	true false

showrange	Boolean value. Whether to show the range of the slider. The range is the "from" and "to" values.	Optional	true false
showcurrentvalue	Boolean value. Whether to show the current value of the slider while it is moving.	Optional	true false
mainbgcolor	Background color of the slider container. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
mainbordercolor	Border color of the slider container. This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #BBBBBB #666666 #666666 #BBBBBB	Optional	#bbb #666 #666 #bbb #BFCFFF #00248F #00248F #BFCFFF
mainborderwidth	Border width of the slider container.	Optional	thin medium thick 1px 2px 5px 10px
pointbgcolor	Background color of the slider point. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
pointbordercolor	Border color of the slider point.	Optional	#bbb #666 #666 #bbb

	This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #BBBBBB #666666 #666666 #BBBBBB		#BFCFFF #00248F #00248F #BFCFFF
pointborderwidth	Border width of the slider point.	Optional	thin medium thick 1px 2px 5px 10px
pointwidth	Width of the slider point in pixels. The value must be an integer value.	Optional	10 20 40 100 300
pointheight	Height of the slider point in pixels. The value must be an integer value.	Optional	10 20 40 100 300
linebgcolor	Background color of the slider line. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
linebordercolor	Border color of the slider line.	Optional	#bbb #666 #666 #bbb

	This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #BBBBBB #666666 #666666 #BBBBBB		#BFCFFF #00248F #00248F #BFCFFF
lineborderwidth	Border width of the slider line.	Optional	thin medium thick 1px 2px 5px 10px
rangefontsize	Font size of the slider range.	Optional	xx-small x-small small medium large x-large xx-large smaller larger 150%
valuebgcolor	Background color of the slider current value which is shown if the "showcurrentvalue" property is set to true. This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
valuebordercolor	Background color of the slider current value which is shown if the "showcurrentvalue" property is set to true.	Optional	#bbb #666 #666 #bbb

	This should be a valid CSS border-color value. You can specify a different color for the top, right, bottom and left border in this sequence. For example: #bbb #666 #666 #bbb		#BFCFFF #00248F #00248F #BFCFFF
valueborderwidth	Border width of the slider current value which is shown if the "showcurrentvalue" property is set to true.	Optional	thin medium thick 1px 2px 5px 10px
valuefontsize	Font size of the slider current value which is shown if the "showcurrentvalue" property is set to true.	Optional	xx-small x-small small medium large x-large xx-large smaller larger 150%
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1	Optional	

	and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

46 STRIPSEL

- Example 372
- Properties 372

The STRIPSEL control is very similar to the TABSTRIP2 control: the user selects one option out of many.

The STRIPSEL control is typically located somewhere at the top of a page, but it can also be positioned anywhere else.

Example

Programming a STRIPSEL control is the same as programming the TABSTRIP2 control - just the rendering of the control differs:



In this example, the STRIPSEL control is the control below the titlebar. For comparison, the TABSTRIP2 control has also been added.

Properties

Basic			
tabstripprop	Name of the adapter parameter that represents the control in the adapter.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property	Optional	left center right

	"textalign" in which you align the control's contained text.		
scrollable	Flag that indicates if the control shows scroll icons on the right upper corner. Default is true	Optional	true false
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
scrolllefttitle	Help text that is displayed if the user moves the mouse of the scroll to left icon.	Optional	
scrolllefttitletextid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
scrollrighttitle	Help text that is displayed if the user moves the mouse of the scroll to right icon.	Optional	
scrollrighttitletextid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
scrollleftimage	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying	Optional	gif jpg jpeg

	"images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
scrollleftimagertl	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
scrollrightimage	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
scrollrightimagertl	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project.	Optional	gif jpg jpeg

	(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

47 SUBCISPAGE2

- Example 378
- Adapter Interface 379
- Session Management 379
- Properties 380

The SUBCISPAGE2 control allows you to embed Application Designer pages such as a NATPAGE into another Application Designer page. You may already have read the section describing the **SUBPAGE** control which allows you to embed any HTML page into an Application Designer page. The differences between the SUBCISPAGE2 control and the SUBPAGE control are:

- With SUBCISPAGE2, you embed an Application Designer page. With SUBPAGE, you embed a normal HTML page.
- Application Designer pages are usually started using the servlet "StartCISPage" which creates an embedding frame in which the Application Designer page is placed. The SUBCISPAGE2 control automatically creates this frame.
- The embedded page is automatically linked to the Application Designer session management. It runs in the same session. This allows the outer page to interact with the embedded page.

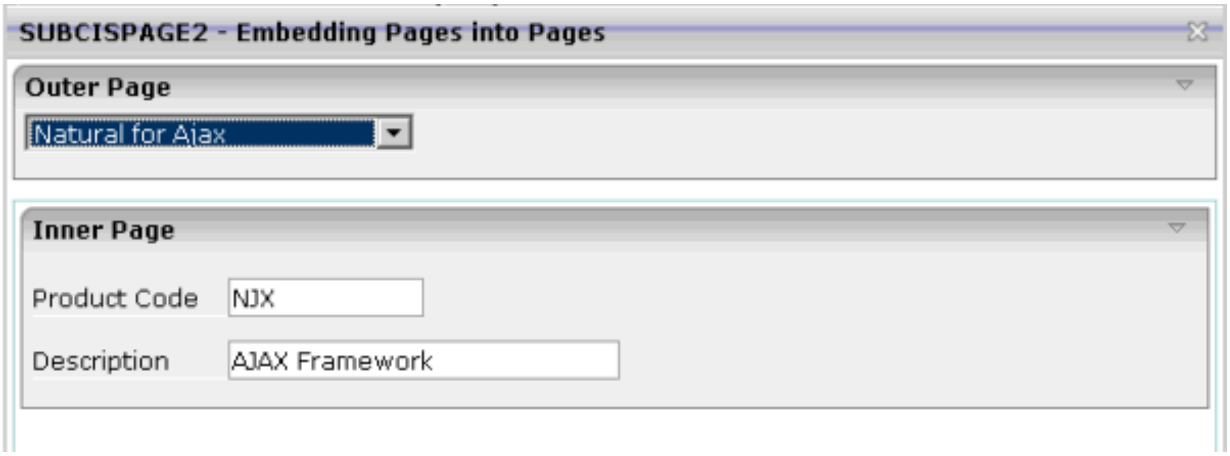
Example

The following is one usage scenario for the SUBCISPAGE2 control:

- Have a page which allows you to select a product from a list.
- Have an embedded page which shows the details of the selected product.

This way, you can modularize complex or large screens. The details can also be delivered by a Natural application which is different from the one which does the search.

Normally, it is not appropriate to use this feature if you just have a few simple fields as shown in the example below. However, this simple example clearly explains how this feature works. You can find it in the Natural for Ajax demos. The outer page allows you to select a product. The inner page shows the product details.



The XML code for the SUBCISPAGE2 control in the outer page is:

```
<rowtable0>
  <itr width="100%">
   <subcispage2 subcispageprop="innerPage" width="100%" height="300" borderwidth="1">
    </subcispage2>
  </itr>
</rowtable0>
```

Adapter Interface

```
1 INNERPAGE
2 CHANGEINDEX (I4)
2 PAGE (A) DYNAMIC
2 PAGEID (A) DYNAMIC
```

Element	Description
CHANGEINDEX	Change this value if the embedded page is to be refreshed. Refreshing means that a server roundtrip will be executed for the embedded page. This allows the corresponding Natural program to send new data to the browser.
PAGE	The URL for opening the embedded page. Example for a NATPAGE: `'/cisnatural/NatLogon.html&xciParameters.natsession=Local&xciParameters.natparamext=stack%3D%28LOGON+SYSEXNJX%3BCTRSBI-P%29'`
PAGEID	For each PAGEID in the same Application Designer subsession, a new Natural session is created. Identical PAGEID elements within a subsession will refer to the same Natural session.

Session Management

A NATPAGE runs in an Application Designer subsession.

NATPAGE pages run in the same Application Designer subsession. Exactly one Natural session is applied to this subsession. If the embedded NATPAGE pages would also use this Natural session, it would not work. Instead, you have to specify a PAGEID for each embedded page. For each different PAGEID within an Application Designer subsession, a new Natural session is opened. All Natural sessions opened during the lifetime of an Application Designer subsession exist as long as the Application Designer subsession exists. For example, if you close the browser, all Natural sessions are closed.

Properties

Basic			
subcispageprop	Name of the adapter parameter that provides the content of the control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	(already explained above)		

borderwidth	Border size of control in pixels. Specify "0" not to render any border at all.	Optional	1 2 3 int-value
withownborder	Default is false. If WITHOWNBORDER is set to true, the subcispage2 control is rendered with its own 3D lookalike border. Set BORDERWIDTH to 0 if WITHOWNBORDER is set to true.	Optional	true false
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50

			int-value
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

48 SUBPAGE

- Properties 384

The SUBPAGE control defines an area in which an HTML page is shown. The URL of the page is not statically defined, but is dynamically controlled by the application.

Due to the browser's capability to embed installed plug-ins, you can use non-HTML objects to be called - and which the browser is able to understand. For example, if you have Microsoft Office installed (or the viewers for Microsoft Office documents) and you pass the name of a Word document as the URL, the Word document will be embedded into the page.

Properties

Basic			
valueprop	Name of the adapter parameter that provides the URL to be displayed inside the SUBPAGE control. Please note: the SUBPAGE control only re-renders its inner content if the URL provided by the property really changes. The SUBPAGE control does not "know" if something changed inside the contained page and that it has to redraw the page. - If you want to refresh the inner page explicitly append some random number to your URL, e.g.: http://...url...?RANDOM=45435. By changing the number the browser will reload the URL.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content.	Sometimes obligatory	100 150 200 250 300

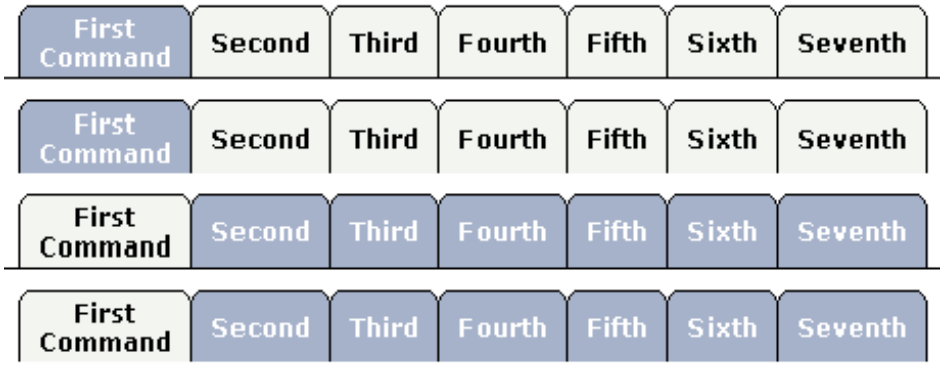
	(B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	(already explained above)		
scrolling	Definition of the scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto yes no
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50

			int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
alwaysreload	When setting to false, the subpage is not reloaded when a page switch is executed, default is true.	Optional	true false
Binding			
valueprop	(already explained above)		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

49 TABSEL

- Adapter Interface 388
- Built-in Events 388
- Properties 389

The TABSEL control looks as shown in the following example:



The number of tabs is dynamically defined at runtime. There are various output options:

- With/without a horizontal line below the control.
- Normal or reverse coloring.

Like the TABSTRIP control, the TABSEL control does not provide internal containers that are switched when selecting tabs. It just represents one tab line.

Adapter Interface

```
1 TABS
2 SELECTEDITEM (I4)
2 TSITEMS (1:*)
3 ID (U) DYNAMIC
3 NAME (U) DYNAMIC
3 TITLE (U) DYNAMIC
```

Built-in Events

value-of-tabselprop.onSelect

Properties

Basic			
tabselprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
bottomborder	If set to "true" then a bottom border is rendered below the tab selection. If set to "false" then no bottom border will be drawn.	Optional	true false
reversecolors	Reverses the color scheme of the TABSEL control.	Optional	true false
leftindent	Inserts a horizontal distance left of the first "tab" and shifts the "tabs" to the right as consequence. The value you may define represents the number of pixels that are inserted.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

50 TABSTRIP2

- Example 392
- Adapter Interface 392
- Built-in Events 392
- Properties 393

The TABSTRIP2 control is used to navigate through certain aspects of your application. The way you navigate depends completely on your implementation.

Example

The control looks as follows:



For each aspect, there is one tab holding a name and an index. The left-most tab holds index 1, the next one 2, etc.

Adapter Interface

```
1 TABS
2 SELINDEX (I4)
2 TSITEMS (1:*)
3 NAME (U) DYNAMIC
```

Built-in Events

value-of-tabstripprop.onSelect

Properties

Basic			
tabstripprop	Name of the adapter parameter that represents the control in the adapter.	Optional	
align	Horizontal alignment of the control's content. Default is "center".	Optional	left center right
scrollable	If set to "true" then small icons will appear on the right border of the control. If the size of the "tabs" is too big and some tabs are cut as consequence then you can use these icons for scrolling left and right.	Optional	true false
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
scrollleftimage	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project.	Optional	gif jpg jpeg

	(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
scrollleftimagertl	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
scrollrightimage	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
scrollrightimagertl	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1)	Optional	

	is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

51 TAGCLOUD

- Example 398
- Adapter Interface 399
- Built-in Events 399
- Properties 399

The TAGCLOUD control represents a collection of tags. A tag is a keyword assigned to an information resource (picture, video clip or others). In a tag cloud, the tags are mainly shown by their popularity.

Example



As you can see, different tags can be added to a tag cloud. They differ by their popularity. The most popular tags are those with a bigger font size.

The XML layout definition is:

```
<itr>
   <tagcloud tagcloudprop="tagCloud"
          width="300" height="350"
          borderstyle="dotted" borderwidth="1px"
          bordercolor="#0000FF" backgroundcolor="#E6E6FA"
          textcolor="#0000FF">
   </tagcloud>
</itr>
```

The tag cloud can be customized by defining a background color.

Adapter Interface

```
1 TAGCLOUD
2 TCLITEM (1:*)
3 ID (U) DYNAMIC
3 POPULARITY (I4)
3 TEXT (U) DYNAMIC
```

The index of the selected tag cloud can easily be determined with an **NJX:EVENTDATA** control.

Built-in Events

value-of-tagcloudprop.onSelect

Properties

Basic			
tagcloudprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container	Optional	100 150 200

	control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 300 250 400 50% 100%
borderstyle	Choose the style the controls border.	Optional	solid double groove dotted dashed inset outset ridge hidden
borderwidth	Border size of control in pixels. Specify "0" not to render any border at all.	Optional	thin medium thick 1px 2px 5px 10px
bordercolor	Sets the border color of the control.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080

			#000000
backgroundcolor	Sets the background color of the control.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
textcolor	Sets the text color of the control.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000

52 TEXT

- Using the max* Properties 404
- Properties 404

The TEXT control represents a multi line text edit control. It represents the value of an adapter parameter.

Using the max* Properties

This section explains when and how to use the following properties:

`maxlength/maxlengthprop`
`maxrows/maxrowsprop`
`maxrowlength/maxrowlengthprop`

The rule of thumb is: `maxlength/maxlengthprop` supports a completely different use case than `maxrows/maxrowsprop` in combination with `maxrowlength/maxrowlengthprop`. Never use all three.

If you simply want to limit the total number of characters that your end users are allowed to enter, use `maxlength/maxlengthprop`.

If you prefer that your end users enter their input line by line with a fixed line length and no automatic word wrapping, use `maxrows/maxrowsprop` in combination with `maxrowlength/maxrowlengthprop`.

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	a width then the rendering result may not represent what you expect.		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will	Optional	string n xs:string

	format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly("true") or not ("false"). Notice that in the Natural code the type for the field is alphanumeric.	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
wrap	Specifies the line wrapping inside the control. By default a line that exceeds the width of the control is broken automatically. You may define this property to not wrap at all ("off") - in this case the text control offers horizontal scroll bars to scroll the text. There are two styles of wrapping "soft" and "hard". The difference between "soft" and "hard" is the way the text is - if changed by the user - passed back to the adapter property: when specifying "soft" then line breaks which are caused by wrapping are not sent to the server, when specifying "hard" then line breaks caused by wrapping are sent as carriage return/ line feed. - Be careful when specifying "hard" as consequence!	Optional	soft hard off

	The wrap attribute is not part of the HTML standard. It depends on the browser if wrap=hard/soft are supported.		
rows	Height of control specified by number of rows. Either define the height by the HEIGHT property or by the ROWS property. Do not specify both! When specifying the height by ROWS then be aware of that the height depends from the font size used inside the control (that is defined in the styles sheet definition).	Optional	
cols	Width of control specified by number of characters. Either define the width by the WIDTH property or by the COLS property. Do not specify both! When specifying the width by COLS then be aware of that the width depends from the font size used inside the control (that is defined in the styles sheet definition).	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
maxlength	Maximum number of characters that a user may enter. This property is not depending on the LENGTH property - please do not get confused by the similar naming. MAXLENGTH has nothing to do with the optical sizing of the control but only with the number of characters you may input.	Optional	5 10 15 20 int-value
maxlengthprop	Name of the adapter parameter that provides the maximum number of characters that a user may enter. Consider to use MAXLENGTH to define this number in a static way.	Optional	
maxrows	Maximum number of rows. No automatic wrapping is done. Don't use this in combination with maxlength/maxlengthprop.	Optional	20 50 100 200

			500 0
maxrowsprop	Name of the adapter parameter that provides the maximum number of rows.	Optional	
maxrowlength	Maximum number of characters in a row. No automatic wrapping is done. Don't this in combination with maxlength/maxlengthprop.	Optional	5 10 15 20 int-value
maxrowlengthprop	Name of the adapter parameter that provides the maximum number of characters in a row.	Optional	
autowordwrap	Can only be used in combination with maxrowlength/maxrowlengthprop and maxrows/maxrowsprop. If set to "true" than when maxrowlength/maxrowlengthprop is reached an automatic word wrap to the next line will be done.	Optional	true false
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
textareastyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied.	Optional	

	Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	
bgcolorprop	Name of an adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is used as background color in the control. The color of the text color is automatically chosen dependent from the background color: for light background colors the text color is black, for dark background colors the color is white. Use FGCOLORPROP to choose the text color on your own.	Optional	
fgcolorprop	Name of an adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is used as text color in the control. The background color is automatically chosen dependent from the text color: for dark text colors the background color is transparent (default), for light text colors the color is black. Use BGCOLORPROP to choose both - the text and background color.	Optional	
scroll	Definition of the scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			

displayprop	(already explained above)		
statusprop	(already explained above)		
titleprop	(already explained above)		
bgcolorprop	(already explained above)		
fgcolorprop	(already explained above)		
maxlengthprop	(already explained above)		
maxrowsprop	(already explained above)		
maxrowlengthprop	(already explained above)		
Online Help			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	(already explained above)		
titletextid	(already explained above)		
titleprop	(already explained above)		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to	Optional	

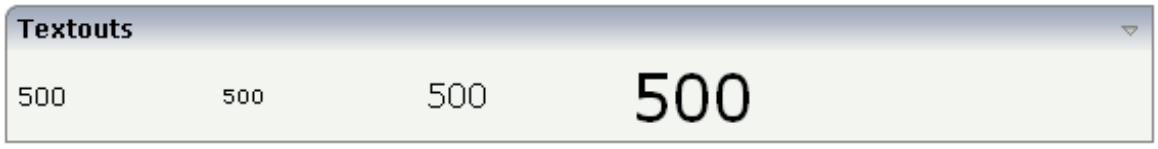
	indicate a generated statusprop variable to which field the statusprop belongs.		
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

53 TEXTOUT

- Example 414
- Properties 414

The TEXTOUT control is used to display plain text. The text is not statically defined (as a label) but is controlled by an adapter property.

Example



The XML layout definition is:

```
<rowarea name="Textouts">
    <itr>
        <textout valueprop="factor1" width="100">
        </textout>
        <textout valueprop="factor1" width="100" textsize="1">
        </textout>
        <textout valueprop="factor1" width="100" textsize="3">
        </textout>
        <textout valueprop="factor1" width="100" textsize="6">
        </textout>
    </itr>
</rowarea>
```

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
nowrap	If the textual content of the control exceeds the size of the control then the browser automatically breaks the line and arranges the text accordingly. You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.	Optional	true false
textsize	The HTML font size of the text. Corresponding to the HTML definition "1" means "smallest" and "6" means "biggest".	Optional	1 2 3 4 5 6

textcolor	Colour of the text. Input a value like "#FF0000".	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	date float int long time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because	Optional	left center right

	the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
bgcolorprop	Name of an adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is used as background color in the control. The color of the text color is automatically chosen dependent from the background color: for light background colors the text color is black, for dark background colors the color is white. Use FGCOLORPROP to choose the text color on your own.	Optional	
fgcolorprop	Name of an adapter parameter that passes back a color value (e.g. "#FF0000" for red color). The color value is	Optional	

	used as text color in the control. The background color is automatically chosen dependent from the text color: for dark text colors the background color is transparent (default), for light text colors the color is black. Use BGCOLORPROP to choose both - the text and background color.		
textoutstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
textoutclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2 VAR3 VAR4
Binding			
valueprop	(already explained above)		
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when the user moves the mouse onto the control.	Optional	

bgcolorprop	(already explained above)		
fgcolorprop	(already explained above)		
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

54 TOGGLE

- Properties 422

The TOGGLE control is used to display and to edit a selection status. In principle, it acts similar to a CHECKBOX control, but it

- allows to define different icon images for the "true" and "false" representations;
- allows being informed when the user presses the CTRL or SHIFT key when clicking the icon. With this information, you can react on a combination of SHIFT and click in a different way than to a normal click or a combination of CTRL and click. This is especially useful inside grid processing when you want to allow the user to do mass selections.

Properties

Basic			
valueprop	Name of the adapter parameter that represents the value of the control.	Obligatory	
trueimage	Image URL that is shown if the corresponding property value is "true".	Obligatory	gif jpg jpeg
falseimage	Image URL that is shown if the corresponding property value is "true".	Obligatory	gif jpg jpeg
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%

height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
partialimage	Image URL that is shown if the corresponding property value is "null".	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0

			1 2 5 10 32767
backgroundclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
Binding			
valueprop	(already explained above)		
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Valid parameter values at runtime: "INVISIBLE", "FOCUS", "FOCUS_NO_SELECT", "ERROR", "ERROR_NO_FOCUS". In responsive controls additionally the values "WARNING" and "SUCCESS" are supported. The value "INVISIBLE" is only supported if the control does not support an INVISIBLEPROP. Use DISPLAYPROP or VISIBLEPROP if available to render the control displayonly/invisible/cleared.	Optional	
shiftmethod	Name of the event that is sent to the adapter when the user clicks on the toggle control and presses the Shift-key the same time.	Optional	
controlmethod	Name of the event that is sent to the adapter when the user clicks on the toggle control and presses the Ctrl-key the same time.	Optional	
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you	Optional	screen server

	want to pass one changed value to all its representation directly after changing the value.		
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

55 CHART

- About the SVG and JPEG Formats 428
- Example 428
- CHART Properties 429
- CHARTCOLUMN Properties 431

The CHART control allows you to present statistics information in a graph. You can choose among several different chart types. Additionally, you can choose the rendering format: SVG or JPEG.

About the SVG and JPEG Formats

When to use which format?

- **SVG**
  SVG is a vector format. You can use it in printable documents because the quality of rendering is scalable.
- **JPEG**
  JPEG is a format for pixel images. It has limited printing quality.

Example

The appearance of the chart can be customized statically at design time or dynamically at runtime. The Natural for Ajax demos contain the following examples, including layouts and corresponding Natural source code:

- **ctrlgraph**
  Shows the settings for the static appearance.
- **ctrlgraph2**
  Shows the settings for the dynamic appearance.

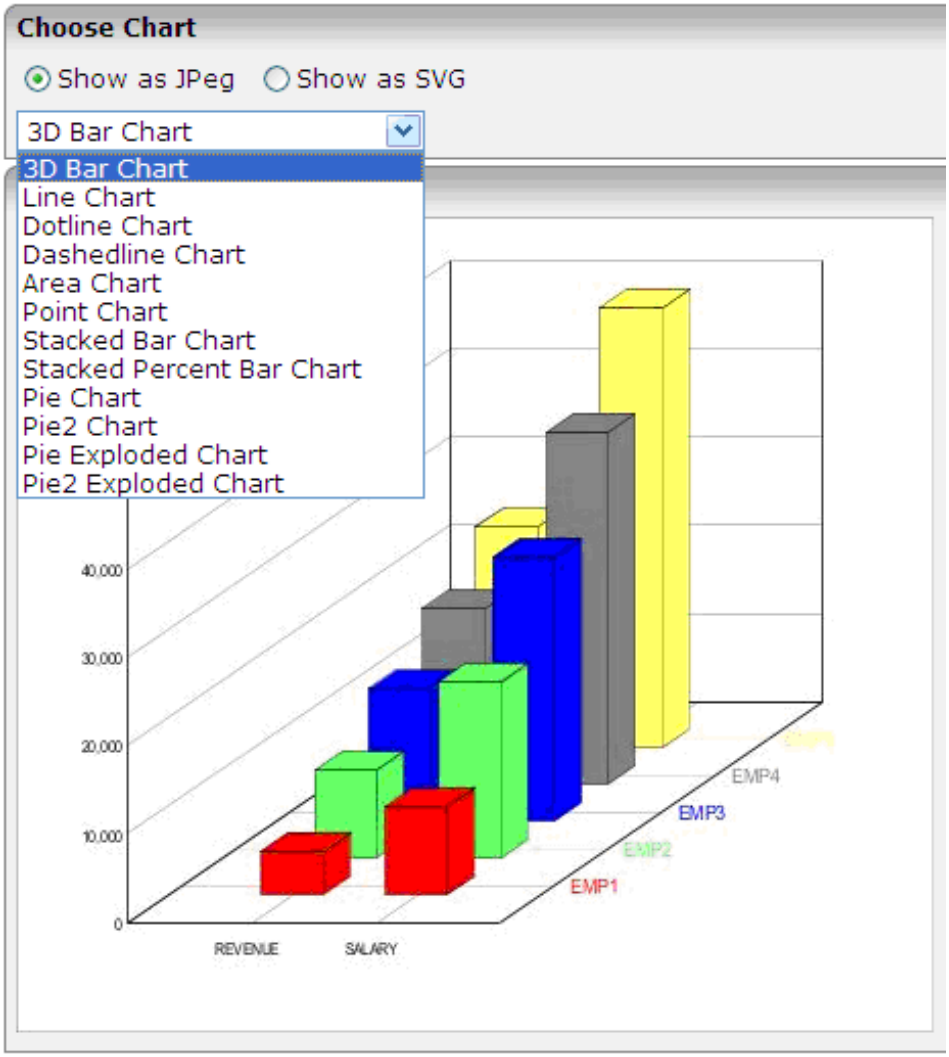


CHART Properties

Basic			
height	Height of the chart. Use pixel definitions only, not percentage definitions.	Optional	1 2 3 int-value
width	Width of chart. Use pixel definitions only, not percentage definitions.	Optional	1 2

			3 int-value
charttype	Type of chart - i.e. whether the output should be rendered as a set of lines, a set of bars, etc.	Optional	bar 3dbar line dotline dashedline area point stacked stacked100 pie pie2 pieexploded pie2exploded
outputformat	Output format: SVG as default. Creation of various image types is supported as well - please open valid values for seeing the list of supported types.	Optional	svg jpeg
colors	Comma separated list of colors. Example: #FF6060;#60FF60;#6060FF. A color is used to render a line. The first color is used for the first line, the second color for the second line and so on. If more lines exist than colors, the colors are used in turns. If no colors are specified a set of default colors is used.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

CHARTCOLUMN Properties

Basic			
property	Each item of a chart represents one line. Each line holds as information (1) the text of the line and (2) multiple key figures that are the values to be rendered as line. E.g. a line may have the values "region,revenue,cost,profit". In this case the "region" is the element passing the text of the line, whereas the other elements are passing the key figure information. For each element you maintain one CHARTCOLUMN item, each one pointing to the data element that passes the value at runtime.	Obligatory	
xaxisproperty	This is an indicator of the attribute "PROPERTY" is holding a text value (then "true") or a key figure value (then "false"). In the example of a line "region,revenue,cost,profit" the corresponding XAXISPROPERTY would be "true,false,false,false".	Optional	true false
title	The text shown for this item as corresponding title on the xaxis. If no title is specified then the property name (see property above) is used as xaxis title.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

56 GOOGLEMAP

- Before You Start 434
- Example 434
- Typical Problems 436
- Properties 436

The GOOGLEMAP control is used to provide for Google Maps support within Application Designer pages.

Before You Start

The GOOGLEMAP control internally makes use of the Google Maps JavaScript API v3. In order to use the GOOGLEMAP control in production, you should obtain an API key from Google. This key is not necessarily required during development. For information on how to obtain an API key, see ***https://developers.google.com/maps/documentation/javascript/tutorial#api_key***.

When using the GOOGLEMAP control, a Google API subpage is needed. Store this page in the registered project directory. You are free in naming the file. The file extension, however, must be *.html*.

The Google API subpage must have the following minimum structure:

```
<html>
  <head>
    <script type="text/javascript" ↩
src="//maps.googleapis.com/maps/api/js?sensor=false"></script>
    <script src="../../HTMLBasedGUI/general/googlemapsscript.js"></script>
  </head>
  <body>
    <div id="map"></div>
  </body>
</html>
```

You can add the API key to the Google API subpage in the following way:

```
<script type="text/javascript" ↩
src="//maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&sensor=false"></script>
```

where the placeholder *YOUR_API_KEY* stands for your API key.

Example

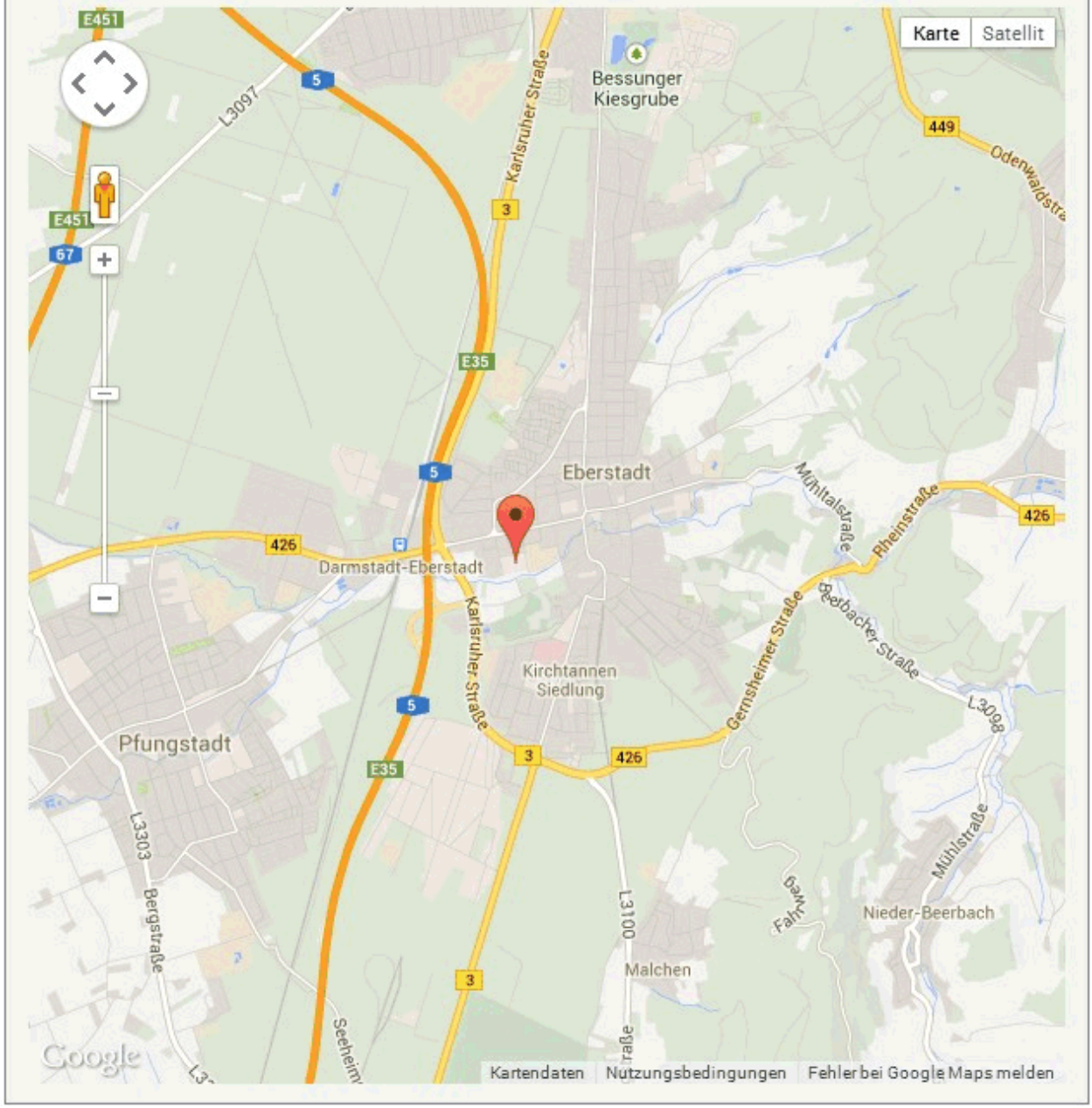
In your Natural program, you can set the address (or latitude and longitude) and optionally the zoom level.

Note: The usage of address or longitude/latitude is mutually exclusive.

For example, when you specify "Uhlandstrasse 12, 64297 Darmstadt, Germany" as the address value, the following map is shown.

Google Map

Demo area



Typical Problems

Map Remains Gray

If you use longitude and latitude for placing the marker on the map, their values may exceed the map top (or bottom) border. If you are able to find the map by scrolling down (or up), then this is the case. Check the values for longitude and latitude in this case.

Properties

Basic			
apikeypagename	Name of the Maps API Key page. Example: mygooglemapsapikey.html. Keep this file within the project directory (directory within the CIS HTML pages are kept). The GOOGLEMAP-control expects this file within certain Javascript includes and content. Have look into chapter "Google Map - Before You Start" within the Developers Guide	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20").	Optional	100 150 200 250 300 250

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	(already explained above)		
mapmode	Lets you toggle between map types (e.g., Map and Satellite)	Optional	1 2
controltype	Lets you toggle between a small and large pan/zoom control	Optional	small large
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value

rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value

57 OPENSTREETMAP and Sub Controls

- Example 440
- Properties for OPENSTREETMAP 440
- Properties for MAPMARKER 443
- Properties for MAPLAYER 446
- Properties for MAPPOINT 447
- Properties for MAPLINE 449
- Properties for MAPPOLYGON 450

The OPENSTREETMAP control supports integrating OpenStreetMap maps into responsive and non-responsive pages.

Optionally, you can use the external geocoder Nominatim with the OPENSTREETMAP control to convert addresses into coordinates and vice versa. When making use of this geocoder, you need to follow the usage policies provided under *https://operations.osmfoundation.org/policies/nominatim/*. Also there is no guarantee of the availability of the Nominatim server.

An OPENSTREETMAP control can have several MAPMARKER controls as sub controls. Each MAPMARKER control describes rendering and data for a group of markers. The marker text, position of the markers and visibility of the markers can be set dynamically at runtime from the Natural programs. Certain events are triggered in the Natural program based on user interactions such as clicking on the map.

Optionally, you can add one or more MAPLAYER controls to an OPENSTRRETMAP. A MAPLAYER is a layer to do area marking on the map. The controls MAPPOINT, MAPLINE, MAPPOLYGON support the marking of points, lines and polygons.

Example

Several examples and corresponding description are provided in the Natural for Ajax demos.

Properties for OPENSTREETMAP

Basic			
withgeocoder	Set this property to true if you want the Ajax framework to use the external geocoder Nominatim to convert addresses to coordinates and vice versa. You need to follow the usage policies of Nominatim https://operations.osmfoundation.org/policies/nominatim/ . Default is false.	Optional	true false
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Optional	100 120 140 160 180 200

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
pixelratio	The ratio between physical pixels and device-independent pixels (dips) on the device.	Optional	1.123456 2

			2.123456
logo	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Optional	
maxzoom	The maximum zoom level (integer).	Optional	1 5 10 20
minzoom	The minimum zoom level (integer).	Optional	1 5 10 20
rotation	The initial rotation for the view in radians (positive rotation clockwise).	Optional	-2.5 -1 1 2.5
zoom	The initial zoom level (integer).	Optional	1 5 10 20
opacity	Opacity (0, 1). Default is 1	Optional	0.2 0.5 1
withoverviewmap	If set to true an icon to show an overview map is added to the lower left corner of the map.	Optional	true

			false
withlocationmarker	If set to true the coordinate specified as location for the map is marked with a default location marker.	Optional	true false
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

Properties for MAPMARKER

Basic			
maxcount	Maximum count of markers for marker group.	Optional	1 2 3 int-value
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height:	Optional	100 150 200

	(A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 300 250 400 50% 100%
style	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
iconurl	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Optional	
styleclasses	CSS style classes separated by a blank.	Optional	
Appearance			

straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
renderaspopover	If set to true the text is rendered as popover when clicking on the icon of the marker. Default is false.	Optional	true false
popoverposition	The position of the popover relative to the icon: top, bottom, left, right.	Optional	top bottom left right
positioning	Defines the positioning with respect to the coordinates of the marker. Possible values are bottom-left, bottom-center, bottom-right, center-left, center-center, center-right, top-left, top-center, and top-right.	Optional	bottom-left bottom-center bottom-right center-center center-right top-left top-center top-right
offset	Offsets in pixels used when positioning the marker. The first value is the horizontal offset. A positive value shifts the overlay right. The second value the vertical offset. A positive value shifts the overlay down. Example: 11	Optional	
insertfirst	If set to true, the markers are inserted first in the in the container.	Optional	true false
autopan	If set to true the map is panned when calling setPosition, so that the marker is entirely visible in the current viewport.	Optional	true false
autopananimation	Animation duration to pan the marker into view.	Optional	500 1000

			2000 6000
autopanmargin	The margin (in pixels) between the marker and the borders of the map when autopanning	Optional	1 2 3 int-value
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

Properties for MAPLAYER

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
zindex	Z-index of the control. If two controls overlap then the one with the higher z-index is drawn in front of the other one.	Optional	1 2 3 int-value
Animation			
withanimation	Set this property to true, if the layer contains animated markers. For optimization purpose, per default animation is switched off.	Optional	true false

Properties for MAPPOINT

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
fillcolor	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
strokecolor	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
strokelinedash	Length and spacing of dashes separated by semicolon. Example: 102	Optional	10;10
strokewidth	The width of the line .	Optional	1 2 3.5
radius	The radius of the point shape. If set to 0, nothing is rendered.	Optional	5 7 10.5
Animation			
animateduration	The duration of the animation in milliseconds.	Optional	500 1000

			2000 6000
animaterepeat	How often the animation is repeated until it is automatically stopped.	Optional	1 2 3 int-value
animatestartradius	The radius with which the animation starts.	Optional	5 7 10.5
animateendradius	The radius with which the animation ends.	Optional	5 7 10.5
animatewidth	The stroke width for the animation.	Optional	1 2 3.5
animatergbcolor	The color for the stroke during animation. If specified as rgb color, for the animation the corresponding rgba color will be used and the opacity is automatically set according to the elapsed animation time.	Optional	rgb(255,0,0) rgb(0,255,0) rgb(0,0,255) rgb(255,255,0)
animatefillrgbcolor	The fill color during animation. If specified as rgb color, for the animation the corresponding rgba color will be used and the opacity is automatically set according to the elapsed animation time.	Optional	rgb(255,0,0) rgb(0,255,0) rgb(0,0,255) rgb(255,255,0)
animateonadd	If set to true, the animation is automatically started when the mappoint is added to a map. This happens when the control is loaded in the browser.	Optional	true false

Properties for MAPLINE

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
strokecolor	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
strokelinedash	Length and spacing of dashes separated by semicolon. Example: 102	Optional	10;10
strokelinecap	Determines what the end of line should look like. Valid values are butt, round, square. Default is round.	Optional	butt round square
strokewidth	The width of the line .	Optional	1 2 3.5
Animation			
animateduration	The duration of the animation in milliseconds.	Optional	500 1000 2000 6000
animaterepeat	How often the animation is repeated until it is automatically stopped.	Optional	1 2 3 int-value

animatedash	In case the stroke line used for the animation should be dashed: Specify the length and spaces of the dashes as comma separated list. Example: 10;2	Optional	10;10
animatewidth	The stroke width for the animation.	Optional	1 2 3.5
animatergbcolor	The color for the stroke during animation. If specified as rgb color, for the animation the corresponding rgba color will be used and the opacity is automatically set according to the elapsed animation time.	Optional	rgb(255,0,0) rgb(0,255,0) rgb(0,0,255) rgb(255,255,0)
animateonadd	If set to true, the animation is automatically started when the mappoint is added to a map. This happens when the control is loaded in the browser.	Optional	true false

Properties for MAPPOLYGON

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
fillcolor	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
strokecolor	Color of the control. Value must follow format "#rrggbb", e.g. #000000 for black.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF

			#808080 #000000
strokelinedash	Length and spacing of dashes separated by semicolon. Example: 102	Optional	10;10
strokewidth	The width of the line .	Optional	1 2 3.5
Animation			
animateduration	The duration of the animation in milliseconds.	Optional	500 1000 2000 6000
animaterepeat	How often the animation is repeated until it is automatically stopped.	Optional	1 2 3 int-value
animatedash	In case the stroke line used for the animation should be dashed: Specify the length and spaces of the dashes as comma separated list. Example: 10;2	Optional	10;10
animatewidth	The stroke width for the animation.	Optional	1 2 3.5
animatergbcolor	The color for the stroke during animation. If specified as rgb color, for the animation the corresponding rgba color will be used and the opacity is automatically set according to the elapsed animation time.	Optional	rgb(255,0,0) rgb(0,255,0) rgb(0,0,255) rgb(255,255,0)
animatefillrgbcolor	The fill color during animation. If specified as rgb color, for the animation the corresponding rgba color will be used and the opacity is automatically set according to the elapsed animation time.	Optional	rgb(255,0,0) rgb(0,255,0) rgb(0,0,255)

			rgb(255,255,0)
animateonadd	If set to true, the animation is automatically started when the mappoint is added to a map. This happens when the control is loaded in the browser.	Optional	true false

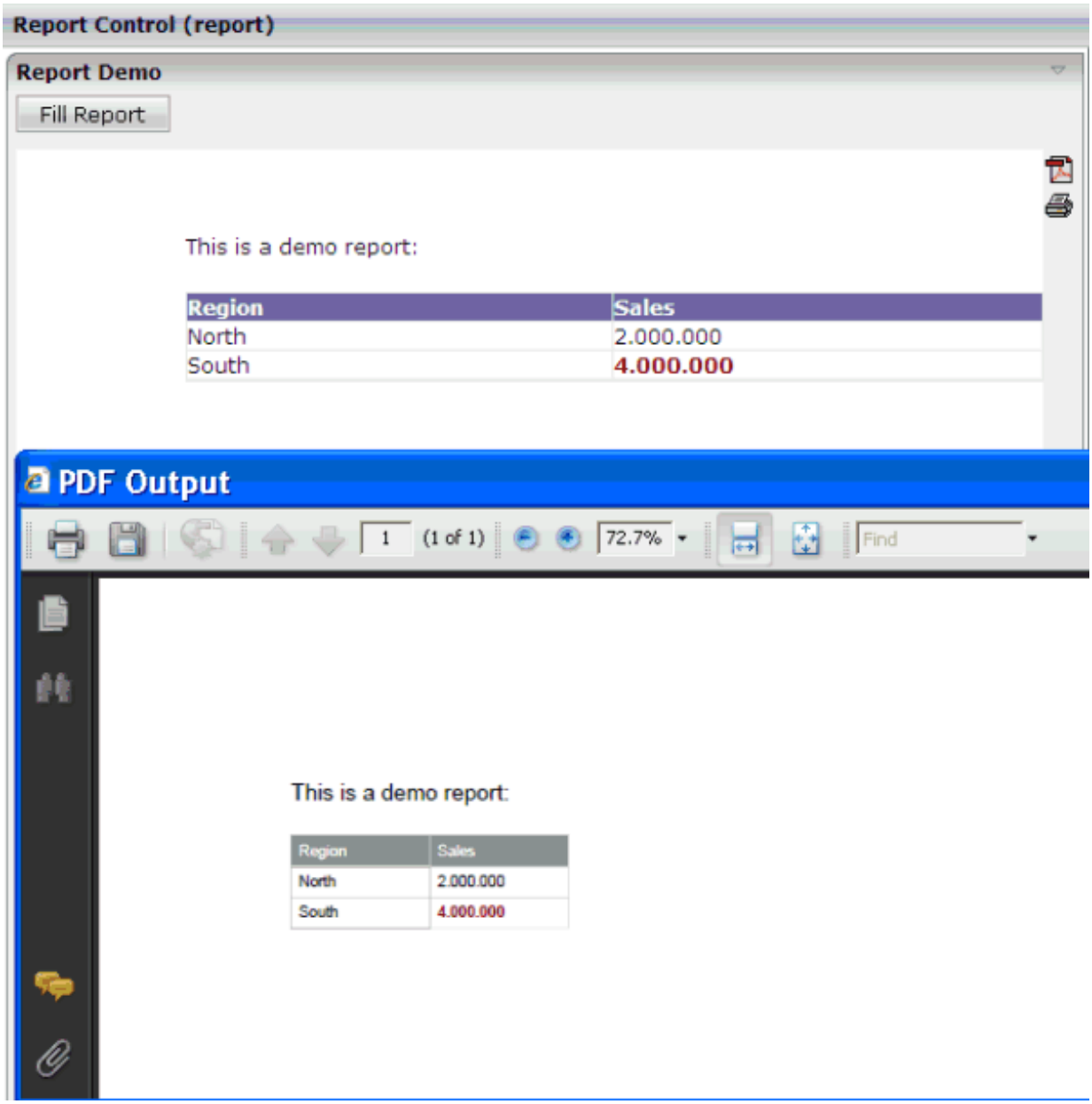
58 REPORT

- Example 454
- Built-in Events 455
- Properties 455

The REPORT control is used to create report output. The report may include text information and table information. It may also have style definitions such as colors and highlighting of certain cells.

Example

The REPORT control provides an automated conversion of the report output into a PDF document, and it allows the user to directly print the report on the client's printer. For more information, see the **ctrlreport** example in the Natural for Ajax demos.



Built-in Events

value-of-reportprop.onGeneratePDF - Assign this event to a button, hot key or other control if you want to trigger the PDF generation under control of the application.

value-of-reportprop.onGeneratePrintVersion - Assign this event to a button, hot key or other control if you want to trigger printing under control of the application.

value-of-reportprop.onUploadPDF - Assign this event to a button, hot key or other control if you want to upload the generated PDF to the Natural server. The PDF content will be added to the **NJX:OBJECTS** cache with "onUploadPDF" as the `CONTENTID`.

Properties

Basic			
reportprop	Name of the server side data representation of the control.	Obligatory	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%
name	A report can be downloaded as pdf file. If you specify a value, this value will be used as file name of the pdf file. Otherwise a default file name is used.	Optional	

nameprop	Name of the adapter parameter that dynamically defines the name of the pdf file. A report can be downloaded as pdf file. Use this property to dynamically define the file name.	Optional	
withwhitespacehandling	Set this property to true if white spaces in some of your report fields - like line breaks and spaces - need to be preserved. This will generate the two additional fields WHITESPACEPRESERVE and TABSPACES in the Natural data structure. You can use these fields to customize the white space handling in the report.	Optional	true false

showpdf	If set to "true" then a PDF icon is rendered in the right top corner of the control. When the user clicks on the icon then the report is automatically rendered as PDF - and the result will show up in a popup window.
	
	Pay attention: if setting this property to "true" then you also have to choose a special constructor when creating the REPORTInfo instance on server side, in which the instance of the model is passed as argument.
	
	Example:
	
	public class XYZAdapter extends Adapter
	
	{
	
	REPORTInfo m_report = new REPORTInfo(this)
	
	...
	
	}
		Optional	true
			
			false

showprintversion	If switched to "true" then a small print icon will appear right from the report area. The print icon opens up a modal popup from which the HTML produced inside the report can be directly sent to the printer. Pay attention: if specifying "true" then the adapter property holding the REPORTInfo object must create the REPORTInfo instance with passing "this" in the constructor.	Optional	true false
showupload	NATPAGE layouts only: If set to "true" then a PDF icon is rendered in the right top corner of the control. When the user clicks on the icon then the report is automatically rendered as PDF and the PDF content is added to the NJX:OBJECTS cache for an upload to the Natural server. The	Optional	true false

	event value-of-reportprop.onUploadPDF is triggered in the Natural application. The Natural application can access the PDF in the NJX:OBJECTS data structure during this event.		
areastyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
areastyleclass	CSS style class used for rendering.	Optional	
fixlayout	The fixlayout property is important for saving rendering performance inside your browser. To become effective it requires to have specified the height and the width (if available as property) of the control. If setting fixlayout to "true" then the control's area is defined as area which is not sized dependent on its content (as normally done with table rendering). Instead the size is predefined from outside without letting the browser "look" into the content of the area. If the content is not fitting into the area then it is cut. You typically use this control if the content of the control's area is flexibly sizable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container. When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible. - The browser as consequence will be much faster in doing its rendering because a screen consists out of "building blocks" with simple to calculate sizes.	Optional	true false

59 REPORT2

- Example 460
- Built-in Events 460
- Properties 460

Like the REPORT control the REPORT2 control is used to create report output. The report may include text information and table information. It may also have style definitions such as colors and highlighting of certain cells. Both controls support the same Adapter interface. Different to the REPORT control, it does not require to additionally render the data in the HTML pages.

Example

The REPORT2 control provides an automated conversion of the report output into a PDF document, and it allows the user to directly print the report on the client's printer or upload the report to Natural. For more information, see corresponding examples in the Natural for Ajax demos.

Built-in Events

value-of-reportprop.onGeneratePDF - Assign this event to a button, hot key or other control if you want to trigger the PDF generation under control of the application.

value-of-reportprop.onGeneratePrintVersion - Assign this event to a button, hot key or other control if you want to trigger printing under control of the application.

value-of-reportprop.onUploadPDF - Assign this event to a button, hot key or other control if you want to upload the generated PDF to the Natural server. The PDF content will be added to the **NJX:OBJECTS** cache with "onUploadPDF" as the `CONTENTID`.

Properties

Basic			
reportprop	Name of the server side data representation of the control.	Obligatory	
name	A report can be downloaded as pdf file. If you specify a value, this value will be used as file name of the pdf file. Otherwise a default file name is used.	Optional	
nameprop	Name of the adapter parameter that dynamically defines the name of the pdf file. A report can be downloaded as pdf file. Use this property to dynamically define the file name.	Optional	
withwhitespacehandling	Set this property to true if white spaces in some of your report fields - like line breaks and spaces - need to be preserved. This will generate the two additional fields WHITESPACEPRESERVE and TABSPACES in the Natural data structure. You can use these fields to customize the white space handling in the report.	Optional	true false

60 TIMER

For detailed information on the **TIMER** control, see ***Non-Visual Controls and Hot Keys***.

61 NJX:BUTTONITEM

- Example 464
- Built-in Events 464
- Properties 464

The NJX:BUTTONITEM control is used to configure the buttons in an **NJX:BUTTONITEMLIST** control. Only one NJX:BUTTONITEM control is needed in an NJX:BUTTONITEMLIST control. This NJX:BUTTONITEM control is used to configure all buttons in the same way.

Example



The XML code for the example looks as follows:

```
<rowarea name="Dynamic Buttonlist">
    <itr>
        <njx:buttonitemlist buttonlistprop="dynbuttons"
        buttoncount="10" hdist="10">
            <njx:buttonitem width="100">
            </njx:buttonitem>
        </njx:buttonitemlist>
    </itr>
</rowarea>
```

Built-in Events

The NJX:BUTTONITEM control behaves like a **BUTTON** control.

Properties

Basic			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL:	Optional	gif jpg jpeg

	(A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
invisiblemode	This property has three possible values: (1) "invisible": the control is not visible without occupying any space. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more. (3)"cleared": the control is not visible but it still occupies space.	Optional	invisible disabled cleared
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to	Optional	100 150 200 250 300 250 400 50%

	have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		100%
imageheight	Pixel height of image inside button.	Optional	
imagewidth	Pixel width of image inside button.	Optional	
textstyle	CSS style definition that is directly passed into the text of this control. With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are: font-weight: bold color: #FF0000	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case	Optional	left center right

	the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.		
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
imagedisabled	URL of image that is displayed if the control is disabled. Use properties VISIBLEPROP and INVISIBLEMODE to disable the control.	Optional	gif jpg jpeg

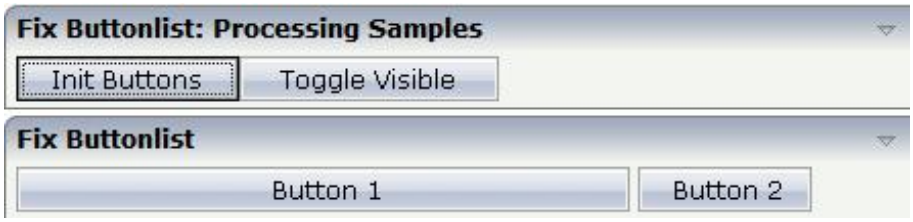
submitbutton	Set this property to true and the button will work as an 'Submitbutton', that is necessary if you want to transfer and/or save form values. i.e. password and username or complete search forms Default value is false. You should only use a 'Submitbutton' if the withformtag option of the pagebody tag is set true.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

62 NJX:BUTTONITEMFIX

- Example 470
- Built-in Events 470
- Properties 471

The NJX:BUTTONITEMFIX control is used to configure the individual buttons in an **NJX:BUTTONITEMLISTFIX** control. For each button in the NJX: BUTTONITEMLISTFIX control, one NJX:BUTTONITEMFIX control is needed.

Example



The XML code for the example looks as follows:

```
<rowarea name="Fix Buttonlist">
    <itr>
        <njx:buttonitemlistfix buttonlistprop="fixbuttons" hdist="4">
            <njx:buttonitemfix method="onButton1"
             invisiblemode="cleared" width="300">
            </njx:buttonitemfix>
            <njx:buttonitemfix method="onButton2"
             invisiblemode="disabled" width="100">
            </njx:buttonitemfix>
        </njx:buttonitemlistfix>
    </itr>
</rowarea>
```

Built-in Events

The NJX:BUTTONITEMFIX control behaves like a **BUTTON** control.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
invisiblemode	This property has three possible values: (1) "invisible": the control is not visible without occupying any space. (2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more. (3)"cleared": the control is not visible but it still occupies space.	Optional	invisible disabled cleared
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of	Optional	100 120 140 160

	container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
imageheight	Pixel height of image inside button.	Optional	
imagewidth	Pixel width of image inside button.	Optional	
textstyle	CSS style definition that is directly passed into the text of this control. With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are: font-weight: bold color: #FF0000	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
buttonstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3 4

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
imagedisabled	URL of image that is displayed if the control is disabled. Use properties VISIBLEPROP and INVISIBLEMODE to disable the control.	Optional	gif jpg jpeg
submitbutton	Set this property to true and the button will work as an 'Submitbutton', that is necessary if you want to transfer and/or save form values. i.e. password and username or complete search forms Default value is false. You should only use a 'Submitbutton' if the withformtag option of the pagebody tag is set true.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Online help			
title	Text that is shown as tooltip for the control.	Optional	

	Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.		
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

63 NJX:BUTTONITEMLIST

- Example 479
- Adapter Interface 479
- Built-in Events 479
- Properties 480

The NJX:BUTTONITEMLIST control is used to arrange buttons in a horizontal line. In contrast to the **NJX:BUTTONITEMLISTFIX** control, the number of buttons in an NJX:BUTTONITEMLIST control can be changed dynamically (up to an upper limit defined at design time), but the layout of the buttons cannot be configured individually. Instead, all buttons in the list are configured with the same layout.

Example



The XML code for the example looks as follows:

```
<rowarea name="Dynamic Buttonlist">
    <itr>
        <njx:buttonitemlist buttonlistprop="dynbuttons"
        buttoncount="10" hdist="10">
            <njx:buttonitem width="100">
            </njx:buttonitem>
        </njx:buttonitemlist>
    </itr>
</rowarea>
```

Adapter Interface

```
1 DYNBUTTONS (1:*)
2 METHOD (A) DYNAMIC
2 NAME (A) DYNAMIC
2 TITLE (A) DYNAMIC
2 VISIBLE (L)
```

Built-in Events

The buttons in the NJX:BUTTONITEMLIST control (NJX:BUTTONITEM controls) behave like **BUTTON** controls.

Properties

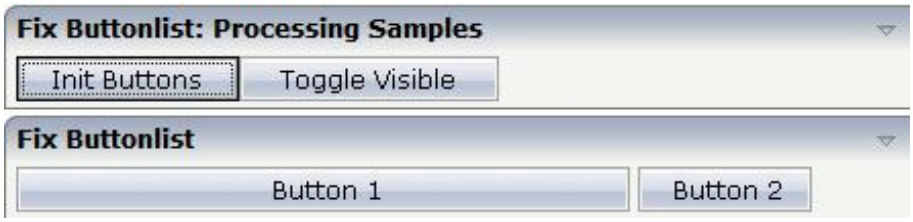
Basic			
buttoncount	Maximum count of buttons in the buttonlist. If no buttoncount is defined then a default of 10 is assigned.	Optional	
hdist	Horizontal distance between the buttons. Can be specified either in pixels or as percentage value. If no width is defined then a default width of 2 pixels is assigned.	Optional	

64 NJX:BUTTONITEMLISTFIX

- Example 482
- Adapter Interface 482
- Built-in Events 483
- Properties 483

The NJX:BUTTONITEMLISTFIX control is used to arrange buttons in a horizontal line. In contrast to the **NJX:BUTTONITEMLIST** control, the number of buttons in an NJX:BUTTONITEMLIST control cannot be changed dynamically, but the layout of the buttons can be configured individually.

Example



The XML code for the example looks as follows:

```
<rowarea name="Fix Buttonlist">
    <itr>
        <njx:buttonitemlistfix buttonlistprop="fixbuttons" hdist="4">
            <njx:buttonitemfix method="onButton1"
             invisiblemode="cleared" width="300">
            </njx:buttonitemfix>
            <njx:buttonitemfix method="onButton2"
             invisiblemode="disabled" width="100">
            </njx:buttonitemfix>
        </njx:buttonitemlistfix>
    </itr>
</rowarea>
```

Adapter Interface

```
1 FIXBUTTONS (1:*)
2 METHOD (A) DYNAMIC
2 NAME (A) DYNAMIC
2 TITLE (A) DYNAMIC
2 VISIBLE (L)
```

Built-in Events

The buttons in the NJX:BUTTONITEMLISTFIX control (NJX:BUTTONITEMFIX controls) behave like **BUTTON** controls.

Properties

Basic			
hdist	Horizontal distance between the buttons. Can be specified either in pixels or as percentage value. If no width is defined then a default width of 2 pixels is assigned.	Optional	

65 NJX:DOCUMENTLINK

- Properties 486

The NJX:DOCUMENTLINK control is used to render text that is dynamically provided by the application through an adapter parameter. The text is rendered as a hyperlink. In a second adapter parameter, the application provides an URL to a document. This URL can refer to documents transported in the data structure of the **NJX:OBJECTS** control. It can also be a normal browser URL for a document which is accessible from within the web application.

When clicking on the hyperlink, the document is opened in a pop-up dialog.



Note: See also ***Documents*** in *Some Common Rules for all Controls*.

Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			

width	(already explained above)		
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
linkstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
linkclass	CSS style class definition that is directly passed into this control. The style class can be either one which is part of the "normal" CIS style sheet files (i.e. the ones that you maintain with the style sheet editor) - or it can be one of an other style sheet file that you may reference via the ADDSTYLESHEET property of the PAGE tag.	Optional	
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right

valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
nowrap	If the textual content of the control exceeds the size of the control then the browser automatically breaks the line and arranges the text accordingly. You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.	Optional	true false
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once	Optional	

	defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

66 NJX:EVENTDATA

- Example 493
- Adapter Interface 494

The NJX:EVENTDATA control supplies additional information related to specific events. With some events, the application needs additional information to handle the event properly. Only one instance of the control needs to be added to the page. This instance provides the event data for all events of other controls on the page that supply additional data. If the page does not contain an instance of the NJX:EVENTDATA control, no additional event data is supplied to the application.

Example



The XML layout definition is:

```
<?xml version="1.0" encoding="UTF-8"?>
<natpage natsource="CTREVD-A" natsinglebyte="true"
xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <titlebar name="Event Data Example">
    </titlebar>
    <pagebody takefullheight="true">
        <rowarea name="Event Data" height="100%">
            <itr height="100%">
                <textgrid2 griddataprop="lines" width="100%"
                 height="100%" selectprop="selected"
                 onclickmethod="lines.onClick">
                    <column name="ID" property="id" width="100">
                    </column>
                    <column name="Last" property="last">
                    </column>
                    <column name="First" property="first">
                    </column>
                </textgrid2>
            </itr>
        </rowarea>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
    <njx:eventdata>
    </njx:eventdata>
</natpage>
```

Adapter Interface

```
1 LINES (1:*)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 SELECTED (L)
1 XCIEVENTDATA
2 XCIINDEX (I4)
```

If a left click is applied to the grid, the index of the line is contained in `XCIEVENTDATA.XCIINDEX`.

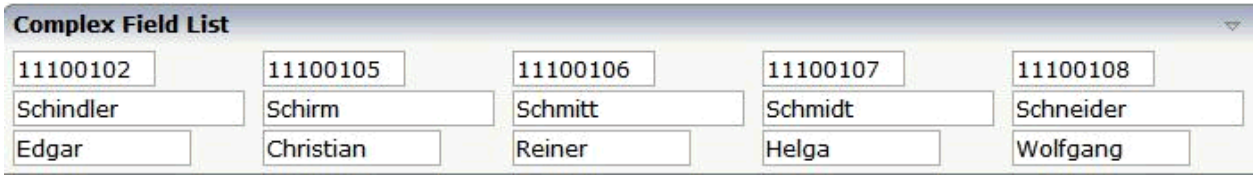
Note that in order to receive the event data, the click event must refer to a specific control. In this example, it must therefore be named `lines.onClick`, not just `onClick`.

67 NJX:FIELDITEM

- Example 497
- Adapter Interface 498
- Built-in Events 498
- Properties 498

The NJX:FIELDITEM control is used to configure the individual fields in an **NJX:FIELDLIST** control in order to create a complex field list. The fields of a complex field list are mapped to a group array in the Natural application. For each field in the NJX:FIELDLIST control, one NJX:FIELDITEM control is needed. The NJX:FIELDITEM controls are used to configure the fields in the list independently.

Example



The XML code for the example looks as follows:

```
<rowarea name="Complex Field List">
    <itr>
        <njx:fieldlist fieldlistprop="columns" fieldcount="5"
         hdist="60">
            <njx:fielditem valueprop="id" width="80"
             invisiblemode="cleared">
            </njx:fielditem>
        </njx:fieldlist>
    </itr>
    <itr>
        <njx:fieldlist fieldlistprop="columns" fieldcount="5"
         hdist="10">
            <njx:fielditem valueprop="last" width="130"
             invisiblemode="invisible">
            </njx:fielditem>
        </njx:fieldlist>
    </itr>
    <itr>
        <njx:fieldlist fieldlistprop="columns" fieldcount="5"
         hdist="40">
            <njx:fielditem valueprop="first" width="100"
             invisiblemode="invisible">
            </njx:fielditem>
        </njx:fieldlist>
    </itr>
</rowarea>
```

Adapter Interface

```
1 COLUMNS (1:*)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 STATUS (A) DYNAMIC
```

For all NJX:FIELDLIST controls that are bound to the same value in `fieldlistprop` (here: `columns`), one common structure array is generated (here: `COLUMNS`).

For each NJX:FIELDITEM control, an element in the structure is generated according to the value bound in `valueprop` (here: `FIRST`, `ID` and `LAST`).

For each occurrence of the structure array, a parameter with the fixed name `STATUS` is generated. This parameter can be used to control the status of the elements in a similar way as it is done with the `statusprop` of the **FIELD** control.

Built-in Events

The fields in the NJX:FIELDLIST control (NJX:FIELDITEM controls or NJX:FIELDVALUE controls) behave like **FIELD** controls.

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
length	Width of FIELD in amount of characters. WIDTH and LENGTH should not be used together. Note that the actual size of the control depends on the font definition if using the LENGTH property.	Optional	5 10 15 20 int-value
maxlength	Maximum number of characters that a user may enter. This property is not depending on the LENGTH property - please do not get confused by the similar naming. MAXLENGTH has nothing to do with the optical sizing of the control but only with the number of characters you may input.	Optional	5 10 15 20 int-value
autotab	If set to true, an automatic tab is executed for fields with a specified MAXLENGTH when the maxlength value is reached. For fields without a MAXLENGTH specified it has no effect. Default is true.	Optional	true false
textalign	Alignment of text inside the control.	Optional	left center right
password	If set to "true", each entered character is displayed as a '*'.	Optional	true false
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false

direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr
uppercase	If "true" then all input is automatically transferred to upper case characters.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows	Optional	1 2

	your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		3 4 5 50 int-value
fieldstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1

			2 5 10 32767
Binding			
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
textidmode	If using property "valuetextprop" then a field knows an id and a text for a certain value. There are three types of display: either both are shown together, separated by an "-" (e.g. "id - text"). Or only text is shown or only the id is shown. If not defined at all then the system's default text id-mode will be chosen. The default mode can be defined as part of the CIS session context.	Optional	
Validation			
datatype	By default, the FIELD control is managing its content as string. By explicitly setting a datatype you can define that the control... ...will check the user input if it reflects the datatype. E.g. if the user inputs "abc" into a field	Optional	date float int long

	with datatype "int" then a corresponding error message will popup when the user leaves the field. ...will format the data coming from the server or coming form the user input: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). In addition value popups are offered for the user automatically for some datatypes: e.g. when specifying datatype "date" the automatically the field provides a calendar input popup. Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.		time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time
validationrules	Contains information used for Data Validation. Use the Validation Rules Editor to make changes!	Optional	
validation	Regular expression against which the content of the field is checked on client side when the user changes the field. If the validation fails then an error message popup up and informs the user about the wrong input.	Optional	[a-zA-Z0-9_.-] {1,}\ \@[a-zA-Z0-9_.-] {1,}\ \.\ \w{2,}\ \d{5} [0-9)(-/+]+
validationuserhint	If a client side validation fails due to wrong user input then an error popup is opened. If you define a hint inside this property then the hint is output to the user in order to tell in which way to input the value. The hint is not language dependent.	Optional	
digits	Number that specifiies how many digits are to be displayed (i.e. digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.	Optional	1 2 3 int-value
decimaldigits	Specifies the number of displayed decimal digits. If using this feature then the DATATYPE property must be set to 'float'.	Optional	1 2

			3 int-value
spinrangemin	An integer value which defines the lower bound of the value range.	Optional	1 2 3 int-value
spinrangemax	An integer value which defines the upper bound of the value range.	Optional	1 2 3 int-value
Valuehelp			
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cursor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false
popupcombowidth	Pixel width of the standard "openIdValueCombo" popup dialog. Default is field width or at least 150 pixel.	Optional	1 2 3 int-value
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image	Optional	gif jpg

	type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		jpeg
touchpadinput	Boolean property that decides if touch pad support is offered for the FIELD control. The default is "false". If switched to "true" then you can input data into the field via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
onlinehelp			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
formula	Contains information used by the Formula Editor. Use the Formula Editor to make changes!	Optional	
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key.	Optional	

	Use the popup help within the Layout Painter to input hot keys.		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

68 NJX:FIELDLIST

- Example 509
- Adapter Interface 510
- Built-in Events 510
- Properties 510

The NJX:FIELDLIST control is used to arrange fields or groups of fields in a horizontal line. The difference of using the NJX:FIELDLIST control instead of individual fields is that the NJX:FIELDLIST control binds the contained fields to an array or array group in the application, while individual fields are bound to individual variables.

Example



The XML code for the example looks as follows:

```
<rowarea name="Complex Field List">
    <itr>
        <njx:fieldlist fieldlistprop="columns" fieldcount="5"
         hdist="60">
            <njx:fielditem valueprop="id" width="80"
             invisiblemode="cleared">
            </njx:fielditem>
        </njx:fieldlist>
    </itr>
    <itr>
        <njx:fieldlist fieldlistprop="columns" fieldcount="5"
         hdist="10">
            <njx:fielditem valueprop="last" width="130"
             invisiblemode="invisible">
            </njx:fielditem>
        </njx:fieldlist>
    </itr>
    <itr>
        <njx:fieldlist fieldlistprop="columns" fieldcount="5"
         hdist="40">
            <njx:fielditem valueprop="first" width="100"
             invisiblemode="invisible">
            </njx:fielditem>
        </njx:fieldlist>
    </itr>
</rowarea>
<rowarea name="Simple Field List">
    <itr>
        <njx:fieldlist fieldlistprop="simple" fieldcount="10">
            <njx:fieldvalue width="50">
            </njx:fieldvalue>
        </njx:fieldlist>
    </itr>
</rowarea>
```

Adapter Interface

```
1 COLUMNS (1:*)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 STATUS (A) DYNAMIC
1 SIMPLE (A/1:*) DYNAMIC
```

For all NJX:FIELDLIST controls that are bound to the same value in `fieldlistprop` (here: `columns`), one common structure array is generated (here: `COLUMNS`).

For each NJX:FIELDITEM control, an element in the structure is generated according to the value bound in `valueprop` (here: `FIRST`, `ID` and `LAST`).

For each occurrence of the structure array, a parameter with the fixed name `STATUS` is generated. This parameter can be used to control the status of the elements in a similar way as it is done with the `statusprop` of the **FIELD** control.

For a simple field list (one that contains an NJX:FIELDVALUE control), a simple array is generated according to the value bound in `valueprop` (here: `SIMPLE`).

Built-in Events

The fields in the NJX:FIELDLIST control (NJX:FIELDITEM controls or NJX:FIELDVALUE controls) behave like **FIELD** controls.

Properties

Basic			
fieldcount	Maximum count of fields in the fieldlist. If no fieldcount is defined then a default of 10 is assigned.	Optional	
hdist	Horizontal distance between the fields Can be specified either in pixels or as percentage value. If no width is defined then a default width of 2 pixels is assigned.	Optional	
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the	Optional	

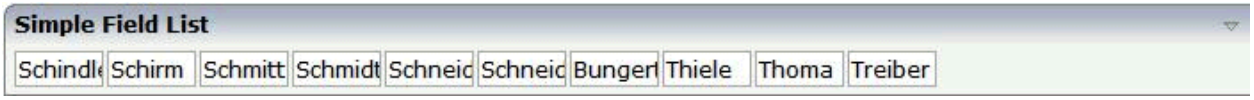
	parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

69 NJX:FIELDVALUE

- Example 515
- Adapter Interface 515
- Built-in Events 515
- Properties 515

The NJX:FIELDVALUE control is used to configure the fields in an **NJX:FIELDLIST** control in order to create a simple field list. The fields of a simple field list are mapped to an array in the Natural application. Only one NJX: FIELDVALUE control is needed in an NJX: FIELDLIST control. This NJX:FIELDVALUE control is used to configure all fields in the list in the same way.

Example



The XML code for the example looks as follows:

```
<rowarea name="Simple Field List">
    <itr>
        <njx:fieldlist fieldlistprop="simple" fieldcount="10">
            <njx:fieldvalue width="50">
            </njx:fieldvalue>
        </njx:fieldlist>
    </itr>
</rowarea>
```

Adapter Interface

```
1 SIMPLE (A/1:*) DYNAMIC
```

For a simple field list (one that contains an NJX:FIELDVALUE control), an array is generated according to the value bound in `valueprop` (here: `SIMPLE`).

Built-in Events

The NJX:FIELDVALUE control behaves like a **FIELD** control.

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content.	Sometimes obligatory	100 120 140 160

	(B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
length	Width of FIELD in amount of characters. WIDTH and LENGTH should not be used together. Note that the actual size of the control depends on the font definition if using the LENGTH property.	Optional	5 10 15 20 int-value
maxlength	Maximum number of characters that a user may enter. This property is not depending on the LENGTH property - please do not get confused by the similar naming. MAXLENGTH has nothing to do with the optical sizing of the control but only with the number of characters you may input.	Optional	5 10 15 20 int-value
autotab	If set to true, an automatic tab is executed for fields with a specified MAXLENGTH when the maxlength value is reached. For fields without a MAXLENGTH specified it has no effect. Default is true.	Optional	true false
textalign	Alignment of text inside the control.	Optional	left center right

password	If set to "true", each entered character is displayed as a '*'.	Optional	true false
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr
uppercase	If "true" then all input is automatically transferred to upper case characters.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3 4 5

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
fieldstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":	Optional	invisible cleared

	(1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.		
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
flush	Flushing behaviour of the input control. By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.	Optional	screen server
textidmode	If using property "valuetextprop" then a field knows an id and a text for a certain value. There are three types of display: either both are shown together, separated by an "-" (e.g. "id - text"). Or only text is shown or only the id is shown. If not defined at all then the system's default text	Optional	

	id-mode will be chosen. The default mode can be defined as part of the CIS session context.		
Validation			
datatype	By default, the FIELD control is managing its content as string. By explicitly setting a datatype you can define that the control... ...will check the user input if it reflects the datatype. E.g. if the user inputs "abc" into a field with datatype "int" then a corresponding error message will popup when the user leaves the field. ...will format the data coming from the server or coming form the user input: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). In addition value popups are offered for the user automatically for some datatypes: e.g. when specifying datatype "date" the automatically the field provides a calendar input popup. Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	date float int long time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time
validationrules	Contains information used for Data Validation. Use the Validation Rules Editor to make changes!	Optional	
validation	Regular expression against which the content of the field is checked on client side when the user changes the field. If the validation fails then an error message popup up and informs the user about the wrong input.	Optional	[a-zA-Z0-9_.-] {1,}\ \ \@[a-zA-Z0-9_.-] {1,}\ \ \.\ \ \w{2,}\ \ \d{5} [0-9)(-/+]+
validationuserhint	If a client side validation fails due to wrong user input then an error popup is opened. If you define a hint inside this property then the hint is output to the user in order to tell in which way to input the value. The hint is not language dependent.	Optional	

digits	Number that specifiies how many digits are to be displayed (i.e. digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.	Optional	1 2 3 int-value
decimaldigits	Specifies the number of displayed decimal digits. If using this feature then the DATATYPE property must be set to 'float'.	Optional	1 2 3 int-value
spinrangemin	An integer value which defines the lower bound of the value range.	Optional	1 2 3 int-value
spinrangemax	An integer value which defines the upper bound of the value range.	Optional	1 2 3 int-value
Valuehelp			
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cursor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false

popupcombowidth	Pixel width of the standard "openIdValueCombo" popup dialog. Default is field width or at least 150 pixel.	Optional	1 2 3 int-value
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	gif jpg jpeg
touchpadinput	Boolean property that decides if touch pad support is offered for the FIELD control. The default is "false". If switched to "true" then you can input data into the field via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
onlinehelp			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
formula	Contains information used by the Formula Editor. Use the Formula Editor to make changes!	Optional	
Hot Keys			

hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

70 NJX:MASHZONE and BMOBILE:MASHZONE

- Before You Start 526
- Example 527
- Adapter Interface 527
- Properties NJX:MASHZONE 528
- Properties BMOBILE:MASHZONE 530
- Individual Data Sources per Session 530

The NJX:MASHZONE and BMOBILE:MASHZONE controls are used to integrate a MashZone NextGen Business Analytics application into a Natural for Ajax page. The data to be displayed is provided by a Natural application. The NJX:MASHZONE control itself consists of a display area for the MashZone application (similar to the area defined with a **SUBPAGE** control) and of a set of data containers representing the data sources of a MashZone data feed.

To use the NJX:MASHZONE or the BMOBILE:MASHZONE control, you must also add an **NJX:OBJECTS** control to your page layout. The NJX:OBJECTS control is used to transport the data objects from the Natural application to the Natural for Ajax server.

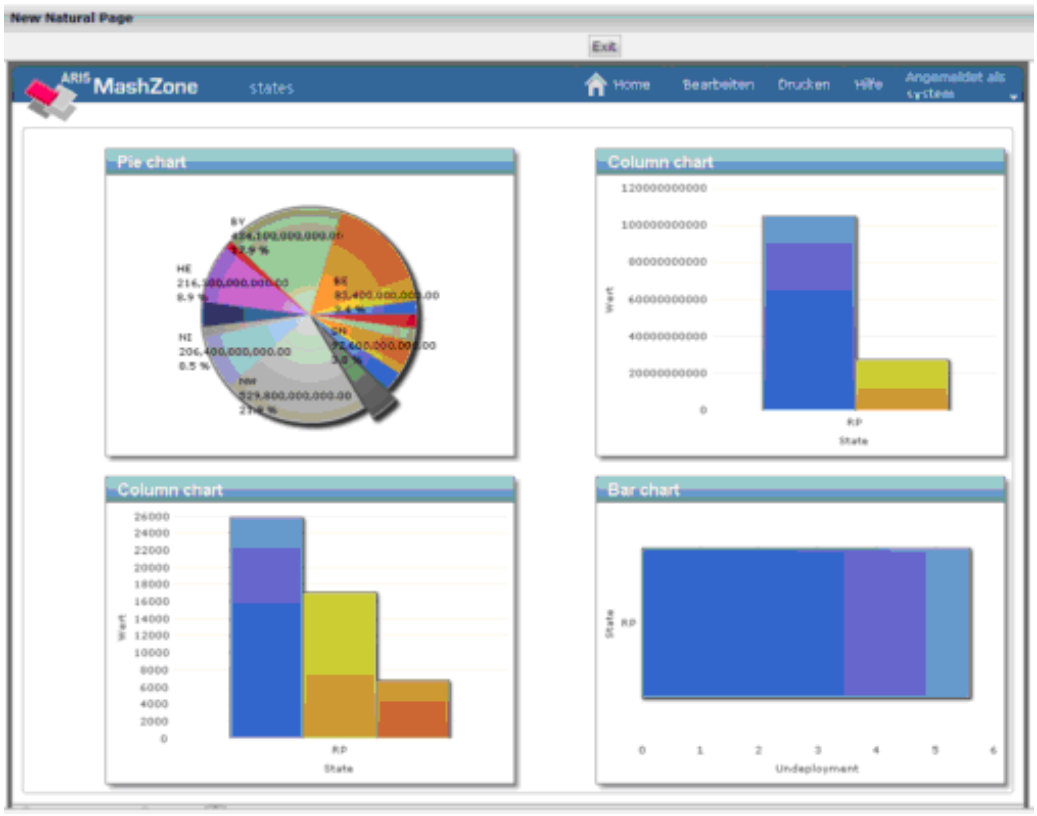
The following topics are covered below:

Before You Start

If you want to use the NJX:MASHZONE or the BMOBILE:MASHZONE control, you have to consider the following:

- The product MashZone NextGen Business Analytics must have been installed beforehand. It is not necessary to install MashZone NextGen on the same host as the Natural for Ajax server. A servlet is provided to access the data via HTTP if different hosts are used.
- The MashZone application that is going to be displayed inside the NJX:MASHZONE control must have been created beforehand. Especially the data source structures on which the data feeds for this MashZone application are based must already be known during application creation time.
- The Natural application used to generate the MashZone data must take care of the correct structure of the data so that the data files generated during runtime correspond to the structures that have been defined for the MashZone application. If, for example, some data source in CSV or XML file format is used in the MashZone application, the Natural application has to provide the data in exactly the same format.

Example



In the NaturalAjaxDemos project of NaturalONE you will find simple executable samples. The used MashZone application is also included in the NaturalAjaxDemos. If you import the sample dashboard into your MashZone installation you can run this sample.

Adapter Interface

```
1 MYMASH
2 DATASOURCE (1:*)
3 FILENAME (A) DYNAMIC
3 OBJECTID (A) DYNAMIC
3 REFRESH (L)
2 URL (A) DYNAMIC
```

The data structure for one NJX:MASHZONE control consists of the following elements:

- `URL` must be filled with the URL of the specific MashZone application.
- For each `DATASOURCE` occurrence, the following elements have to be filled:

Element	Description
FILENAME	The name of the file that is used as data source by the MashZone application.
OBJECTID	The "nat:" URL obtained when filling the NJX:OBJECTS structures with the BLOB contents.
REFRESH	Indicates whether the BLOB content is to be copied into the file during the next page update. This should be set to TRUE during the initial page load in order to fill the file accordingly. Possible values: TRUE or FALSE.

Properties NJX:MASHZONE

Most control properties are identical with the properties of the same names of the **SUBPAGE** control.

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50%	Sometimes obligatory	100 150 200 250 300 250 400 50%

	then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
height	(already explained above)		
scrolling	Definition of the scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto yes no
pagestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but	Optional	1 2

	you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		3 4 5 50 int-value
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

Properties BMOBILE:MASHZONE

Same as for NJX:MASHZONE. See property table above for details.

Individual Data Sources per Session

The `MashZoneDataSource` servlet can be used as data source URL in the MashZone dashboard to access session-specific data of a Natural Ajax application. This servlet is automatically configured in the *web.xml* of the Natural Ajax product. The `MashZoneDataSource` servlet supports the following URL parameters: `SESSIONID` and `NAME`. The `SESSIONID` is the Natural Ajax session ID.

Add an `NJX:REQUESTCONTEXT` to your page layout - then you can simply pass on the value `XCIREQUESTCONTEXT.CISSESSIONID` as URL parameter in the MashZone URL. The servlet is used as data source URL in the dashboard of the MashZone application.

The mapping between the URL parameters of the MashZone URL to the URL parameters of the data sources is done in the MashZone application. See the description in the NaturalAjaxDemos of NaturalONE for details on how to establish this mapping.

71 NJX:NJXFILEDOWNLOAD

- Example 535
- Adapter Interface 535
- Properties 535

The NJX:NJXFILEDOWNLOAD control is used to add a link to your page layout for downloading files from the Natural server to the client.

To use the NJX:NJXFILEDOWNLOAD control, you must also add an **NJX:OBJECTS** control to your page layout. You can link to all files available in the NJX:OBJECTS cache. The link is directly passed to the browser. To force the browser to download a file instead of opening it, you append the parameter `DOWNLOAD=true` to your link (for example, "nat:mydoc?DOWNLOAD=true"). The `CONTENTID` of the file in the NJX:OBJECTS cache will be used as a suggestion for the name of the downloaded file. You can also refer to files outside of the NJX:OBJECTS cache by using a normal browser link. In this case, the `DOWNLOAD` parameter is not evaluated by the Natural for Ajax framework.

To trigger a download from another control (for example, from a BUTTON control), you have to use the **SUBPAGE** control instead. See the Natural for Ajax demos for an example.

The following topics are covered below:



Note: See also ***Documents*** in *Some Common Rules for all Controls*.

Example



The XML layout definition is:

```
<itr>
   <label name="Download Link" width="100">
   </label>
   <njx:njxfiledownload  valueprop="mydownload" >
   </njx:njxfiledownload>
</itr>
```

When you choose the **Download from Natural** link in this example, the file download dialog from your operating system appears.

Adapter Interface

```
1 MYDOWNLOAD
2 LINKPROP (A) DYNAMIC
2 NAMEPROP (A) DYNAMIC
```

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width	Optional	100 120 140 160 180 200 50% 100%

	of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
ihtmlstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

72 NJX:NJXFILEUPLOAD2

- Example: File Upload to Natural 539
- Adapter Interface: Upload to Natural Server 539
- Adapter Interface: Upload to Filesystem 539
- Built-in Events 540
- Properties 540

The NJX:NJXFILEUPLOAD2 control is used to upload files from the client to the Natural server or to a designated directory on the web application server running on Windows or Linux. Per default, the files are uploaded to the Natural server as BLOBs. For uploading to a designated directory on the web application server (Windows or Linux), set the design time property filestoragemode="Filesystem". The rendering is identical to the FILEUPLOAD2 control.

When uploading to Natural you must also add an NJX:OBJECTS control to your page layout because the uploaded file will be added to the NJX:OBJECTS data structure. Once a file is uploaded, the Natural application can refer to the file via the CONTENTID of the uploaded object. When uploading files to the filesystem, the generated adapter interface provides additional parameters that control the destination of the uploaded file:

- destpath (required) - Specifies the target directory on the web application server. The path must be absolute and must already exist.
- subpath (optional) - Specifies a subdirectory below destpath. If the directory does not exist, it is created automatically. For security reasons, only valid subdirectories are accepted; directory traversal patterns such as ../ are rejected.

The following topics are covered below:



Note: See also ***Documents*** in *Some Common Rules for all Controls*.

Example: File Upload to Natural



The XML layout definition is:

```
<itr>
   <njx:njxfileupload2 width="400" valueprop="myupload"
        withsubmitbutton="true" submitbuttonname="Upload to Natural">
   </njx:njxfileupload2>
</itr>
```

Adapter Interface: Upload to Natural Server

```
1 MYUPLOAD
2 CEXT (A) DYNAMIC
2 CNAME (A) DYNAMIC
2 CONTENTID (A) DYNAMIC
2 CPATH (A) DYNAMIC
```

When uploading a file, the client-side file name, extension and path are provided in the `CNAME`, `CEXT` and `CPATH` fields, respectively. The Natural application can optionally specify a `CONTENTID` for the `NJX:OBJECTS` data structure. If `CONTENTID` is not specified, a concatenation of file name and extension is automatically set as the `CONTENTID` value.

Adapter Interface: Upload to Filesystem

```
1 MYUPLOAD
2 DESTNAME (A) DYNAMIC
2 DESTPATH (A) DYNAMIC
2 RESULTCODE (I4)
2 SUBPATH (A) DYNAMIC
```

When uploading a file, the file name is provided in `DESTNAME`. The `RESULTCODE` field indicates the outcome of the operation.

Result Codes

- 1 - The destination path is not an absolute path.
- 2 - The destination path does not exist.
- 3 - The specified subpath is invalid.
- 4 - A file with the same name already exists in the destination location.
- 5 - An I/O error occurred while writing the file.

Built-in Events

When you click the **Upload to Natural** button in the above example, the *value-of-valueprop*.onUploadFinished event is triggered in the Natural application. When this event is triggered, either the CNAME, CEXT, CPATH and CONTENTID fields are filled, and the file content is provided in the data structure of the NJX:OBJECTS control, or the DESTNAME and RESULTCODE fields are filled, depending on whether you chose upload to Natural or to the filesystem.

When you click the **Upload** button, the corresponding valueprop.onUploadFinished event is triggered in the Natural application. The information returned with this event depends on the selected storage mode:

- **Upload to Natural** - The fields CNAME, CEXT, CPATH, and CONTENTID are populated. In addition, the file content is made available through the data structure of the NJX:OBJECTS control, which can be accessed using the returned CONTENTID.
- **Upload to Filesystem** - The fields DESTNAME and RESULTCODE are populated. DESTNAME contains the name of the stored file, while RESULTCODE indicates whether the upload completed successfully or provides an error code describing the reason for a failure.

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100").	Optional	100 120 140 160 180 200

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		50% 100%
withsubmitbutton	If set to "TRUE" adds an additional button to the control to start the file upload.	Optional	true false
submitbuttonname	The name of the submit button in case WITSUBMITBUTTON is set to "true".	Optional	
submitbuttontextid	"Textid" for the name of the submitbutton if WITHSUBMITBUTTON is set to "true". The "textid" is translated into a corresponding string at runtime. Do not specify WITHSUBMITBUTTONNAME if specifying WITHSUBMITBUTTONID.	Optional	
maxfilesize	Maximum size of the file. The unit is specified in the filesizeunit property.	Optional	1 2 3 int-value
filesizeunit	The unit for the maxfilesize property. Default is bytes.	Optional	bytes KB MB GB
filestoragemode	Per default the file is uploaded to your Natural program as a BLOB. Chose the value Filessystem if you prefer to upload the file to the filesystem on the web application server instead.	Optional	BLOB Filesystem
overwrite	Applies only if filestoragemode Filesystem is set. Per default already existing files are overwritten during the upload. Set this property to false to prevent overwriting existing files. If a file with the same name already exists, the upload will fail and return an error.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			

invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible disabled cleared
Appearance			
invisiblemode	(already explained above)		
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
darkbackground	Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used to integrate light coloured controls into darker container areas.	Optional	true false
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	

73 NJX:NJXVARIABLE

- Example 544
- Properties 544

The NJX:NJXVARIABLE control is used in Natural Map Converter templates in order to define a placeholder that is replaced during map conversion. For further information, see *Templates* in the section *Customizing the Map Conversion Process* of the *Application Modernization* part.

Example

The Map Converter template `NATPAGE_TEMPLATE` contains a variable `MAPROOT` that receives the result of the map conversion process. As a result, the converted Natural map content is placed into the `pagebody` of the resulting page layout.

```
<?xml version="1.0" encoding="UTF-8"?>
<natpage xmlns:njx="http://www.softwareag.com/njx/njxMapConverter"
natsource="$$NATSOURCE$$" natsinglebyte="true">
    <titlebar name="$$TITLEVAR$$" align="center">
    </titlebar>
    <pagebody>
        <njx:njxvariable name="MAPROOT"/>
    </pagebody>
    <statusbar withdistance="false"/>
</natpage>
```

Properties

Basic			
name	The name of the variable.	Optional	

IV Working with Grids

This part shows you how to deal with grids. Working with grids is as simple as working with singular properties because the grid management adapts seamlessly into the normal processing of the Application Designer environment.

The information provided in this part is organized under the following headings:

Basics

TEXTGRID2

TEXTGRIDSSS2 - TEXTGRID2 with Server-Side Scrolling

ROWTABLEAREA2 - The Flexible Control Grid

ROWTABLEAREA3 - The Array Grid

FLEXLINE - Flexible Columns in Control Grids

MGDGRID - Managing the Grid

GRIDCOLHEADER - Flexible Column Headers

Styling Grids

74 Basics

It is quite simple: “normal” controls refer to an adapter and are bound to adapter parameters. Grid controls refer to an adapter as well - but are bound to a group array. Each array element provides group elements to access its content.

Two types of grid controls are available:

- The TEXTGRID2 control is a control that displays grid data - but does not allow any change to the data. You can select grid rows and colorize them in different ways. Change the order of columns dynamically and sort columns by clicking into the title row of the grid.

  There is a TEXTGRIDSSS2 control that is a certain variant of the TEXTGRID2 control.
- The ROWTABLEAREA2 is a container that internally allows you to use any normal control to be embedded inside a grid. Therefore, you can place normal FIELD controls, CHECKBOX controls etc. inside the ROWTABLEAREA2 container.

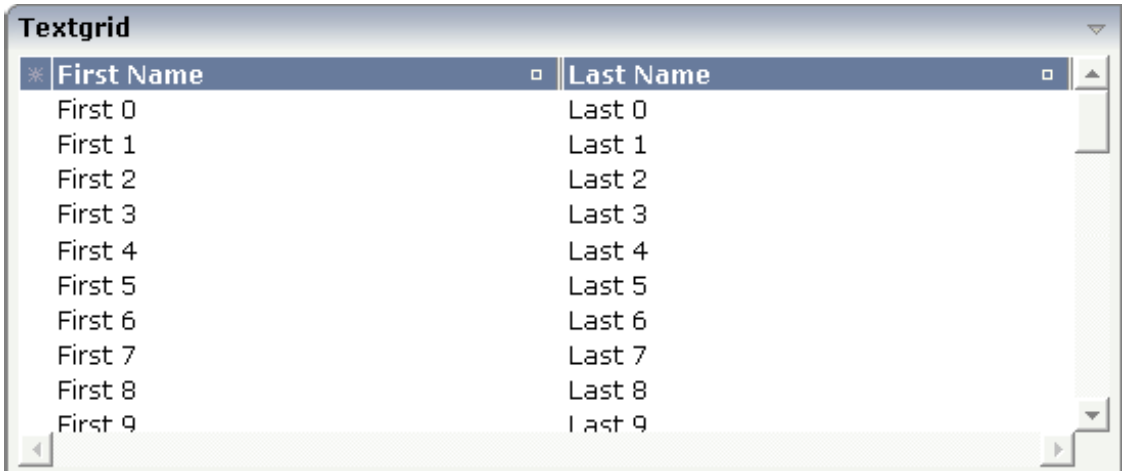
Use the TEXTGRID2 controls for displaying and selecting data. Use ROWTABLEAREA2 for entering data inside a grid.

75 TEXTGRID2

- A Simple Example 550
- Adapter Interface for TEXTGRID2 551
- Selecting Rows in a TEXTGRID2 551
- TEXTGRID2 Properties 552
- COLUMN Properties 558
- Dynamic Setting of Text Styles in TEXTGRID2 562
- CSVCOLUMN Properties 564

A Simple Example

The following example shows a TEXTGRID2 control:



There are two columns which hold data. There is one column at the very left which displays a selection icon - in addition to a yellow background for a selected line. Even and odd lines are displayed in slightly different colors. At the very right of each title column, there is a symbol which indicates the sorting status; if you double-click on this symbol, the column is sorted first in ascending direction and, when clicking again, in descending direction. Change the sequence of columns by dragging the title of a column and dropping it on another column's title. Depending from where you drop, the column is either moved left or right.

The asterisk in the upper left corner of the grid is used to select/deselect all lines in the grid. The behavior depends on the setting of the `singleselect` property which determines whether multiple lines can be selected in the grid (default) or whether only one line can be selected:

- **Multiple Line Selection Mode**
  When you choose the asterisk for the first time, all lines are selected. When you choose the asterisk a second time, all lines are deselected.
- **Single Line Selection Mode**
  When you choose the asterisk (no matter how often), an existing selected line is deselected.

The XML layout definition is:

```
<rowarea name="Textgrid">
    <itr takefullwidth="true" fixlayout="true">
        <textgrid2 griddataprop="lines" width="100%" height="200" ↩
selectprop="selected"
                   hscroll="true">
            <column name="First Name" property="firstName" width="50%">
            </column>
            <column name="Last Name" property="lastName" width="50%">
            </column>
        </textgrid2>
    </itr>
    <vdist height="5">
    </vdist>
</rowarea>
```

The TEXTGRID2 definition is bound to a grid data property `lines`.

Inside the TEXTGRID2 control definition there are two columns. These columns are bound to the properties `firstName` and `lastName`.

Adapter Interface for TEXTGRID2

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
```

Selecting Rows in a TEXTGRID2

Maybe you wonder why there is a `selected` field in the adapter parameter data area of the previous example.

This field is required for indicating which lines are currently selected and which are not. Each line which is displayed in the TEXTGRID2 control is represented in the adapter by an array occurrence of the array `LINES`. Therefore, the selection status of the grid (which lines are selected and which lines are not) is mirrored by the corresponding `selected` field of each array occurrence.

TEXTGRID2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the grid in the adapter.	Obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 150 200 250 300 250 400 50% 100%

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Selection			
selectableprop	Name of the adapter parameter that specifies whether a row in the grid is selectable (=true) or not (=false). The default is selectable.	Optional	
selectprop	Name of the adapter parameter that is used to mark if an individual row of the text grid is selected. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
singleselect	If set to "true" then only one row can be selected inside the text grid. - If set to "false" then multiple lines can be selected by using Ctrl- and Shift-key during mouse selection. Default is "false".	Optional	true false
singleselectprop	Name of an adapter parameter that dynamically defines whether SINGLESELECT is true or false.	Optional	
onclickmethod	Name of the event that is sent to the adapter when the user selects a row. In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user selects a row by a double click. In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".	Optional	
withselectioncolumn	When defining a SELECTPROP property then automatically a selection column is added as first left column of the grid. Inside the column an icon indicates if a row is currently selected. Set this property to "false" in order to avoid the selection column.	Optional	true false
withselectioncolumnicon	Flag that indicates whether the selection column shows a "select all" icon on top. Default is true.	Optional	true false
fgselect	if switched to true then an additional "graying" of selected lines will be activated. Switch this property to "true" if you have coloured textgrid cells: the	Optional	true false

	selection colour will not override the colour of each cell, as consequence you require an additional effect in order to make the user see which row is selected.		
focusedprop	Name of an adapter parameter that is used to mark if an individual row of the text grid should receive the focus. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
focusable	Set this to TRUE if you want the browser to treat the grid like a focusable element (like buttons and input elements) when the page is loaded. Example usage: When your grid is the first focusable element the browser will automatically set the focus on the grid when the page is loaded.	Optional	true false
gridfocusedprop	Name of the adapter parameter that indicates if the grid should receive the focus. It is automatically reset after the server-roundtrip.	Optional	
cursormgtprop	Only use this if you would like to reset the keyboard cursor from your Natural program. Name of the adapter parameter that sets -1 in case the keyboard cursor should be reset after this server roundtrip.	Optional	
Right Mouse Button			
oncontextmenumethod	Name of the event that is sent to the adapter when the user clicks with the right mouse button onto an empty area of the grid.	Optional	
singleselectcontextmenu	With SHIFT and CTRL key the user can select multiple lines (use property SINGLESELECT to suppress this feature). Use this property to ensure that the context menu is requested only for a single line. Default is "false".	Optional	true false noselection
enabledefaultcontextmenu	Use this property to enable the default context menu of the browser within the textgrid. Please note: do not enable the browser's context menu if your application itself provides for a context menu. Default is "false".	Optional	true false
Appearance			
width	(already explained above)		
height	(already explained above)		
minapparentrows	Number of rows that are displayed independent of the size of the server side collection.	Optional	1 2

			3 int-value
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Sometimes obligatory	auto scroll hidden
withtitlerow	If defined as "false" then no top title row is shown. "True" is default.	Optional	true false
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
personalizable	If defined to "false" then no re-arranging of columns is offered to the user. Default is "true". This means: if using COLUMN controls inside the grid definition then the user can re-arrange the sequence of columns by dragging and dropping them within the top title row.	Optional	true false

stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
stylevariantprop	Name of the adapter parameter which dynamically defines the STYLEVARIANT value at runtime.	Optional	VAR1 VAR2
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "scroll".	Optional	auto scroll hidden
withrollover	The textgrid controls provide for a so called "roll over" effect. The row that is currently below the mouse pointer is highlighted in a certain way. Use this property to disable the roll over effect (Default is TRUE).	Optional	true false
fixedcolumnsizes	When switching the FIXEDCOLUMNSIZES property to value "true" then internally the grid is arranged in	Optional	true

	a way that the area always determines its size out of the width specification of the COLUMN controls. The browser does not look into the column contents in order to try to optimise the size of the area - but always follows the width that you define.		false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%). Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.	Optional	1 2 3 int-value
disablecolumnresizing	Flag that indicates if the user can change the width of the grid columns. Default is false.	Optional	true false
disablecolumnmoving	Flag that indicates if the user can change the order of grid columns. Default is false.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Drag And Drop			
draginfoprop	Name of the row item property that passes back the line's "drag info". When using this attribute the grid lines can be dragged onto "drop targets" (e.g. DROPICON control). The dragged line is identified by its "drag info". Use any string/information applicable.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name	Optional	

	(in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
Deprecated			
directselectevent	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	ondblclick onclick
directselectmethod	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	

COLUMN Properties

The COLUMN tag is the typical tag that is placed inside a TEXTGRID2 definition. The COLUMN definition defines a column with its binding to a property of the collection elements.

Tip: If you set the property `headernowrap="false"`, you usually have to increase the height of the header in the style sheet of your layout page. You can do this in the Style Sheet Editor: Go to the **Style Details** tab, expand the tree for TEXTGRID and then adjust the `height` value for `TEXTGRIDCellHeaderUnsorted`.

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
property	Property of the row item object that represents the column's content. The content typically is straight text but can also be "complex HTML".	Obligatory	
width	Width of the control. There are two possibilities to define the width: (A) Pixel sizing: just input a number value (e.g. "100"). (B) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element (textgrid2, textgridsss2) of the control properly defines a width this control can reference.	Obligatory	100 120 140 160 180 200 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
datatype	By default, the control is managing its content as string. By explicitly setting a datatype you can define that the control will format the data coming from the server: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings). Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.	Optional	date float int long time timestamp color xs:decimal xs:double

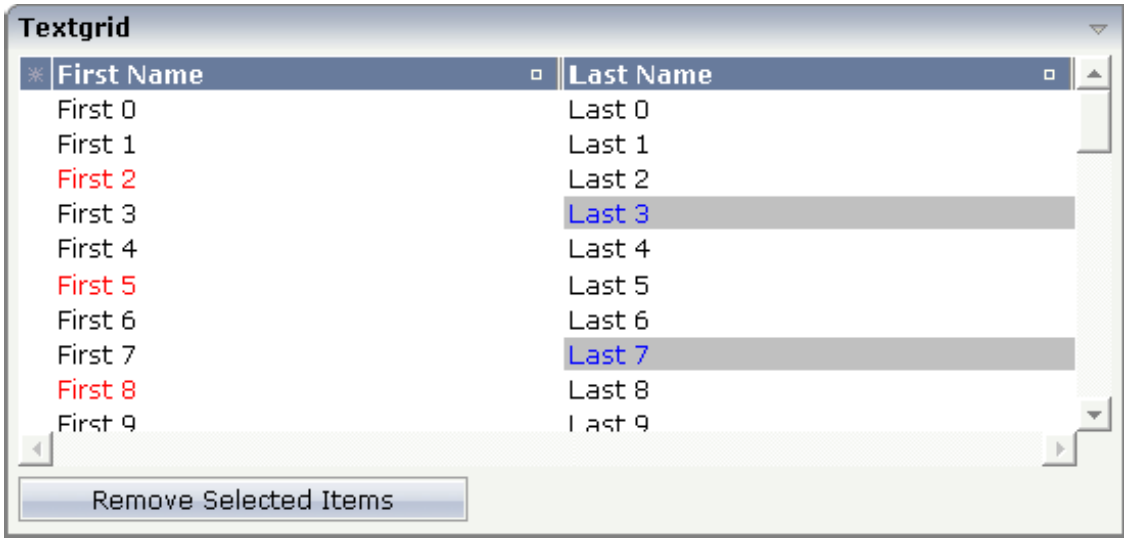
			xs:date xs:dateTime xs:time
align	Horizontal alignment of the control's content. Default is "center".	Optional	left center right
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
convertspaces	If switched to "true" then all spaces inside the text that is rendered into the column are converted to non breakable spaces. Use this option if you have "meaningful" spaces inside the values you return from the server adapter object, e.g. if outputting some ASCII protocol inside a column.	Optional	true false
cuttextline	If switched to "false" then the content of the column is broken if it exceeds the column's width definition. Default is "true" i.e. if the content is too big for the column cell then it is cut.	Optional	true false
withsorticon	Flag that indicates if a small sort indicator is shown within the right corner of the control. Default is TRUE.	Optional	true false
headerimage	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".	Optional	
headernowrap	The textual content of the header is not wrapped automatically. No line break will be performed	Optional	true

	automatically by the browser. If you want the text of the header to be wrapped, set the value to "false".		false
Binding			
property	(already explained above)		
textstyleprop	Name of the adapter parameter that provides a style-string that is used for rendering the column's content. As consequence you can individually assign a CSS-style to each cell of your text grid.	Optional	
textclassprop	Name of the adapter parameter that provides a style class to be used for rendering the content. You can set up a limited number of style classes inside your style sheet definition - and dynamically reference them per grid cell.	Optional	
imageprop	Name of the adapter parameter that provides an image URL. The image is rendered at the very left of the column's area - in front of the text (PROPERTY property definition).	Optional	
linkmethod	Name of the event that is sent to the adapter if user clicks the column's text.	Optional	
celllinkmethodprop	Name of the row item property that passes back the name of a method or null. If the method name is not null then the corresponding column (cells) will show the text as method link. On click the provided row item cell method is called.	Optional	
celltitleprop	Name of the adapter parameter that provides the tooltip of this cell.	Optional	
Online help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
sorttitle	Text that is shown as tooltip for the sort indicator. Either input text by using this SORTTITLE property - or use the SORTTITLETEXTID in order to define a language dependent literal.	Optional	
sorttitletextid	Text ID that is passed to the multi language management - representing the tooltip text for the sort indicator.	Optional	
celltitleprop	(already explained above)		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the	Optional	

	control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

Dynamic Setting of Text Styles in TEXTGRID2

The example from the previous sections will now be enhanced in order to demonstrate how to control the style of cells inside a TEXTGRID2 control dynamically:



Some of the cells in the TEXTGRID2 control are rendered with a different style than the normal one. Each COLUMN definition has the property `textstyleprop`:

```
<rowarea name="Textgrid">
    <itr takefullwidth="true" fixlayout="true">
        <textgrid2 griddataprop="lines" width="100%" height="200" ↩
selectprop="selected"
                   hscroll="true">
            <column name="First Name" property="firstName" width="50%"
                    textstyleprop="firstNameStyle">
            </column>
            <column name="Last Name" property="lastname" width="50%"
                    textstyleprop="lastNameStyle">
            </column>
        </textgrid2>
    </itr>
    <vdist height="5">
    </vdist>
    <itr>
        <button name="Remove Selected Items" method="onRemoveSelectedItems">
        </button>
    </itr>
</rowarea>
```

CSVCOLUMN Properties

There are situations in which the number and the format of the columns of a text grid cannot be defined in a fixed way inside the layout definition. The column type CSVCOLUMN allows you to dynamically define columns of a grid by your program.

The properties of the CSVCOLUMN control are:

Basic			
titlesprop	Name of adapter property provding a semicolon-separated string containing the titles to be displayed. Example for a value that is passed back by the property: "First Name;Last Name;Street""	Obligatory	
valuesprop	Name of row item property that passes back the content of the cells - as semicolon-separated string.	Obligatory	
widthsprop	Name of adapter property provding a semicolon-separated string containing the widths of the columns to be displayed. Example for a value that is passed back by the property: "100;200;100%""	Obligatory	
alignsprop	Name of adapter property provding a semicolon-separated string containing the horizontal alignment of the columns to be displayed. Example for a value that is passed back by the property: "left\"center;right""	Sometimes obligatory	
backgroundsprop	Name of adapter property provding a semicolon-separated string containing the background color of the columns to be displayed. Example for a value that is passed back by the property: "\"#C0C0C0;#FF0000""	Optional	
proprefsprop	Name of adapter property provding a semicolon-separated string containing the row item properties that are internally used to build up the value string. The property names are used for sorting: if the user invoke the sorting of the grid by clicking on the corresponding icons inside the title cell then this column needs to be associated with an internal property that is used for sorting. Example: "firstName\"lastName;street""	Optional	

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
straighttext	If the text of the control contains HTML tags then these are by default interpreted by the browser. specifying STRAIGHTTEXT as "true" means that the browser will directly render the characters without HTML interpretation. Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".	Optional	true false
cuttextline	If switched to "false" then the content of the column is broken if it exceeds the column's width definition. Default is "true" i.e. if the content is too big for the column cell then it is cut.	Optional	true false
headernowrap	The textual content of the header is not wrapped automatically. No line break will be performed automatically by the browser. If you want the text of the header to be wrapped, set the value to "false".	Optional	true false
withgridcolheaders	Flag that indicates if the user can resize column widths and re-order columns by drag and drop. Default is false. If set to true the corresponding adapter program must register as "column change event" listener. Use method TEXTGRIDCollection.registerGridColHeaderChangeListener for that.	Optional	true false
Binding			
textstyleprop	Name of the adapter parameter that provides a style-string that is used for rendering the column's content. As consequence you can individually assign a CSS-style to each cell of your text grid.	Optional	
textclassprop	Name of the adapter parameter that provides a style class to be used for rendering the content. You can set up a limited number of style classes inside your style sheet definition - and dynamically reference them per grid cell.	Optional	
straighttextprop	Name of the adapter parameter which dynamically defines whether STRAIGHTTEXT is true or false.	Optional	
sorttitlesprop	Name of adapter property provding a semicolon-separated string containing the titles to be displayed. Example for a value that is passed back by the property: "Click here to sort column First Name\" Click here to sort column Last Name; Click here to sort column Street""	Optional	
tooltiptitlesprop	Name of adapter property provding a semicolon-separated string containing the tooltip tip texts to be displayed when the mouse is moved over the column headers.	Optional	

linkmethodsprop	Name of the property of the row item object that passes back (comma separated) names of row item methods. The corresponding columns will show the text as method links. On click the provided row item method is called.	Optional	
celllinkmethodsprop	Name of the row item property that passes back (comma separated) names of cell methods. The corresponding columns (cells) will show the text as method links. On click the provided row item cell method is called.	Optional	
celltooltiptitleprop	Name of the property of the row item object that passes back (comma separated) tool tip titles. The titles will show up if the user is moving slowly the mouse over the grid cells.	Optional	
imageprop	Name of the property of the row item object that passes back (comma separated) image URLs. The URL must either be an absolute URL or a relative URL.	Optional	
headerimageprop	Name of the adapter parameter which provides a comma separated list of image URLs. The images are applied to the header.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

76 TEXTGRIDSSS2 - TEXTGRID2 with Server-Side Scrolling

- Performance Considerations 568
- Example 568
- Adapter Interface 570
- Using Server-Side Scrolling 570
- Using Server-Side Sorting 571
- Setting the Client-Side Loading Behavior 571
- TEXTGRIDSSS2 Properties 571

The TEXTGRIDSSS2 control is a variant of the **TEXTGRID2** control which is explained in the previous section. "SSS" is the abbreviation for "server-side scrolling". For a general explanation of server-side scrolling, see *Server-Side Scrolling and Sorting*.

Performance Considerations

The TEXTGRID2 control fetches all items belonging to the grid and renders them according to its layout definition. If there are more items available than the grid can display, a vertical scroll bar is displayed and you can scroll through the list.

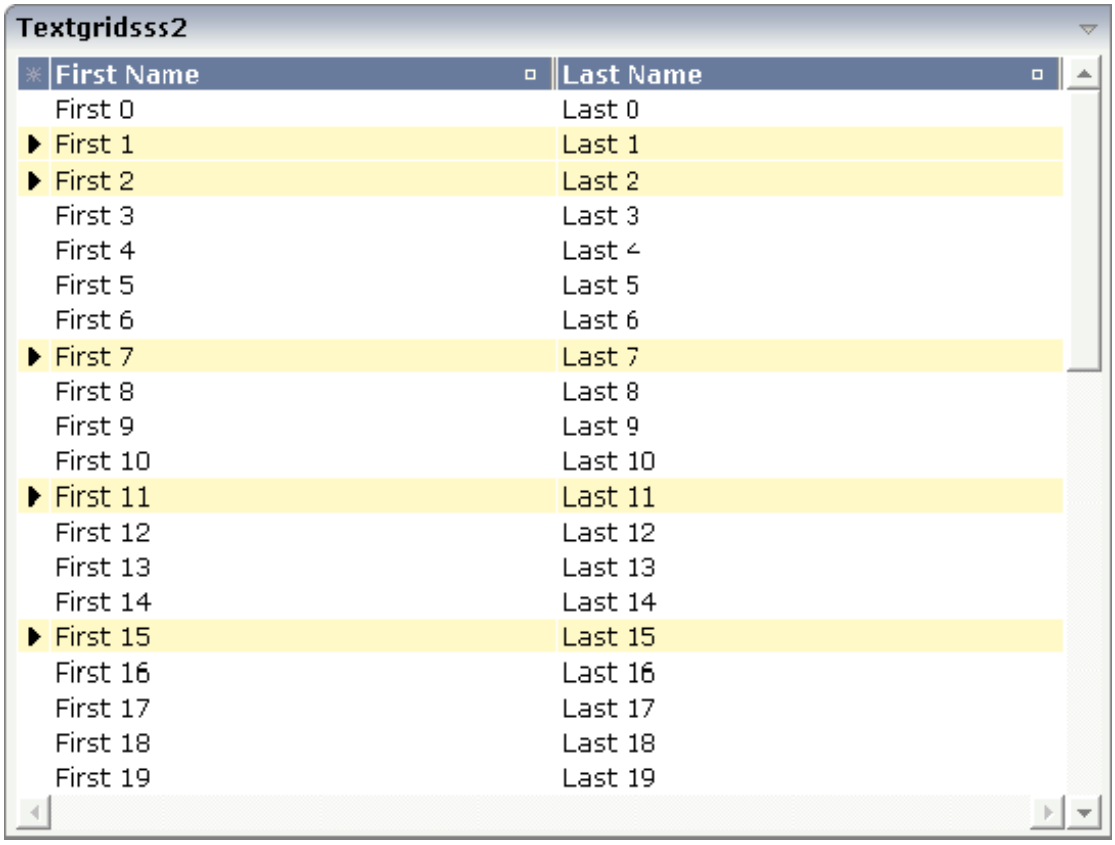
From scrolling perspective, this is very effective - the browser is very fast when scrolling is needed. But there are two disadvantages, especially for long lists:

- All the data that are to be displayed inside the grid must be available on the client side. Therefore, the data must be transferred from the server to the client at least one time. Imagine you have a grid of 10,000 lines: even if Application Designer transfers only “net data” and even if this happens in “delta transfer mode”, it must be transferred.
- In addition, the grid must be built completely in order to allow fast scrolling. This means - taking the above example - that 10,000 lines have to be rendered before the grid can be displayed. Table rendering is time-consuming and needs a lot of the client's CPU performance.

Consequence: text grids of the TEXTGRID2 control are easy to use, but they have their limitations in terms of scalability. You should use it only if a limited amount of information is to be displayed.

Example

The TEXTGRIDSSS2 is very similar to the TEXTGRID2 control. However, some special behavior has been built in. The main differences are “in the background”. The TEXTGRIDSSS2 control only receives the data of the visible items. In this example, only the data of the first 20 items are returned and rendered. When scrolling down, the next 20 items are fetched and rendered. This means: the control requests always the data which are currently displayed.



Consequence: every scrolling step requires an interaction with the server. However, only a small amount of data - which is visible - is requested, not the data of all available items. The performance of the grid does not change with the number of items which are available. There is no time difference in rendering a text grid containing 100 or 10,000 items.

The layout definition is:

```
<rowarea name="textgridsss2">
    <itr>
        <textgridsss2 griddataprop="lines" rowcount="20" width="100%"
                      selectprop="selected" singleselect="false" hscroll="true"
                      directselectmethod="onDirectSelection"
                      directselectevent="ondblClick">
            <column name="First Name" property="firstname" width="50%">
            </column>
            <column name="Last Name" property="lastname" width="50%">
            </column>
        </textgridsss2>
    </itr>
</rowarea>
```

The application can decide at runtime whether to use this feature or not. For information on how this can be controlled, see *Server-Side Scrolling and Sorting*.

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
1 LINESINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)
```

The parameters are nearly the same as for the TEXTGRID2 control. In addition, there is a `LINESINFO` structure. This structure is used to control the server-side scrolling and the server-side sorting.

Using Server-Side Scrolling

In the adapter parameters that represent the TEXTGRIDSSS2 control in the application, there are three parameters that control the server-side scrolling:

- `TOPINDEX`
- `ROWCOUNT`
- `SIZE`

In `TOPINDEX` and `ROWCOUNT`, the application receives the information how many items it should deliver to the page with the next scroll event and with which item the delivered amount should start.

In `SIZE`, the application returns the total number of items available. The client uses this information to set up the scroll bar correctly.

Using Server-Side Sorting

In the adapter parameters that represent the TEXTGRIDSSS2 control in the application, there is a substructure that controls the server-side sorting: `SORTPROPS`. With the information in this structure, the client tells the application by which sort criteria and in which order the client expects the items to be sorted.

Setting the Client-Side Loading Behavior

As an alternative to server-side scrolling, you can customize the client-side loading behavior. Setting the property `onloadbehaviour="collection"` activates performance-optimized client-side scrolling in the client. As with the TEXTGRID2 control, the application must pass all items to the client at the beginning. But other than with the TEXTGRID2 control, these items are not immediately rendered in the grid. Instead, the client caches the items and only renders the items that are currently visible. If you have a limited number of items, the TEXTGRIDSSS2 control thus combines the advantage of the easy-to-use TEXTGRID2 control with better performance. For a really large number of items, however, server-side scrolling is still the best solution.



Note: If you set `onloadbehaviour="collection"` for a grid control at design time, it is not possible to perform Natural server-side scrolling for this grid at runtime.

Sometimes, the number of items is not yet known at design-time. In such a case, you can set `onloadbehaviour="collectionorblock"`. Up to a defined number of items, the behavior is identical to `onloadbehaviour="collection"`. The maximum number of items for which client-side scrolling is to be done can be specified in the *cisconfig.xml* file. For a higher number of items, server-side scrolling is done. Switching between client-side scrolling and server-side scrolling is done automatically. You need not program anything to make it work.

TEXTGRIDSSS2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the grid in the adapter.	Obligatory	
rowcount	Number of rows that are rendered inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property:	Obligatory	

	If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that are defined as ROWCOUNT value. If a HEIGHT value is defined in addition (e.g. as percentage value "100%") then the number of rows depend on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that are picked from the server. You should define this value in a way so that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too high ("100") because this causes more communication traffic and more rendering effort inside the browser.		
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Obligatory	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20").	Optional	100 150 200 250 300 250 400

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		50% 100%
onloadbehaviour	DEPRECATED: Loading behaviour of the items into the client. "block" (=default). Only "block" is supported´. All other settings are automatically mapped to "block".	Optional	block collection collectionorblock
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Selection			
selectableprop	Name of the adapter parameter that specifies whether a row in the grid is selectable (=true) or not (=false). The default is selectable.	Optional	
selectprop	Name of the adapter parameter that is used to mark if an individual row of the text grid is selected. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
singleselect	If set to "true" then only one row can be selected inside the text grid. - If set to "false" then multiple lines can be selected by using Ctrl- and Shift-key during mouse selection. Default is "false".	Optional	true false
singleselectprop	Name of an adapter parameter that dynamically defines whether SINGLESELECT is true or false.	Optional	
onclickmethod	Name of the event that is sent to the adapter when the user selects a row. In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user selects a row by a double click.	Optional	

	In the event handler you can find the selected rows by iterating through the rows and finding out which one's selected element is set to "true".		
dblclickonreturn	If set to TRUE, the RETURN key will trigger the same method as an double click.	Optional	
withselectioncolumn	When defining a SELECTPROP property then automatically a selection column is added as first left column of the grid. Inside the column an icon indicates if a row is currently selected. Set this property to "false" in order to avoid the selection column.	Optional	true false
withselectioncolumnicon	Flag that indicates whether the selection column shows a "select all" icon on top. Default is true.	Optional	true false
fgselect	if switched to true then an additional "graying" of selected lines will be activated. Switch this property to "true" if you have coloured textgrid cells: the selection colour will not override the colour of each cell, as consequence you require an additional effect in order to make the user see which row is selected.	Optional	true false
focusedprop	Name of an adapter parameter that is used to mark if an individual row of the text grid should receive the focus. If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.	Optional	
focusable	Set this to TRUE if you want the browser to treat the grid like a focusable element (like buttons and input elements) when the page is loaded. Example usage: When your grid is the first focusable element the browser will automatically set the focus on the grid when the page is loaded.	Optional	true false
gridfocusedprop	Name of the adapter parameter that indicates if the grid should receive the focus. It is automatically reset after the server-roundtrip.	Optional	
cursormgtprop	Only use this if you would like to reset the keyboard cursor from your Natural program. Name of the adapter parameter that sets -1 in case the keyboard cursor should be reset after this server roundtrip.	Optional	
Right Mouse Button			

oncontextmenumethod	Name of the event that is sent to the adapter when the user clicks with the right mouse button onto an empty area of the grid.	Optional	
singleselectcontextmenu	With SHIFT and CTRL key the user can select multiple lines (use property SINGLESELECT to suppress this feature). Use this property to ensure that the context menu is requested only for a single line. Default is "false".	Optional	true false noselection
enabledefaultcontextmenu	Use this property to enable the default context menu of the browser within the textgrid. Please note: do not enable the browser's context menu if your application itself provides for a context menu. Default is "false".	Optional	true false
Appearance			
width	(already explained above)		
height	(already explained above)		
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "hidden".	Optional	auto scroll hidden
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "scroll".	Optional	auto scroll hidden
touchpadinput	Boolean property that decides if touch pad support is offered for the TEXTGRID control. The default is "false". If switched to "true" then you can scroll the grid via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
withtitlerow	If defined as "false" then no top title row is shown.	Optional	true false

	"True" is default.		
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
personalizable	If defined to "false" then no re-arranging of columns is offered to the user. Default is "true". This means: if using COLUMN controls inside the grid definition then the user can re-arrange the sequence of columns by dragging and dropping them within the top title row.	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	

stylevariantprop	Name of the adapter parameter which dynamically defines the STYLEVARIANT value at runtime.	Optional	VAR1 VAR2
backgroundstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
withrollover	The textgrid controls provide for a so called "roll over" effect. The row that is currently below the mouse pointer is highlighted in a certain way. Use this property to disable the roll over effect (Default is TRUE).	Optional	true false
fixedcolumnsizes	When switching the FIXEDCOLUMNSIZES property to value "true" then internally the grid is arranged in a way that the area always determines its size out of the width specification of the COLUMN controls. The browser does not look into the column contents in order to try to optimise the size of the area - but always follows the width that you define.	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%).	Optional	1 2 3 int-value

	Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.		
minapparentrows	Minimum number of apparent rows. Insert a valid number to make sure that (e.g. 10) rows are shown for sure.	Optional	1 2 3 int-value
disablecolumnresizing	Flag that indicates if the user can change the width of the grid columns. Default is false.	Optional	true false
disablecolumnmoving	Flag that indicates if the user can change the order of grid columns. Default is false.	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
removeemptylines	Only supported in NATPAGE layouts. When setting to TRUE, empty lines will be removed from the item array passed to NJX. The reduced array will be rendered in the browser. An empty line is a line which contains only empty alphanumeric fields or numeric fields which are 0.	Optional	true false
withsliderfreeze	Setting this to "true" prevents unwanted slider jumps while scrolling up/down in a grid with a huge number of lines (for example 20000).	Optional	true false
Drag And Drop			
draginfoprop	Name of the row item property that passes back the line's "drag info". When using this attribute the grid lines can be dragged onto "drop targets"	Optional	

	(e.g. DROPICON control). The dragged line is identified by its "drag info". Use any string/information applicable.		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	
Deprecated			
directselectmethod	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	
directselectevent	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	ondblclick onclick
showemptylines	If set to false, no empty line will be rendered. By default empty lines are shown.	Optional	true false

Inside the TEXTGRIDSSS2 definitions, COLUMN tags are also used to define its content. There is no difference in COLUMN tag usage between TEXTGRIDSSS2 and TEXTGRID2 definition.

77 ROWTABLEAREA2 - The Flexible Control Grid

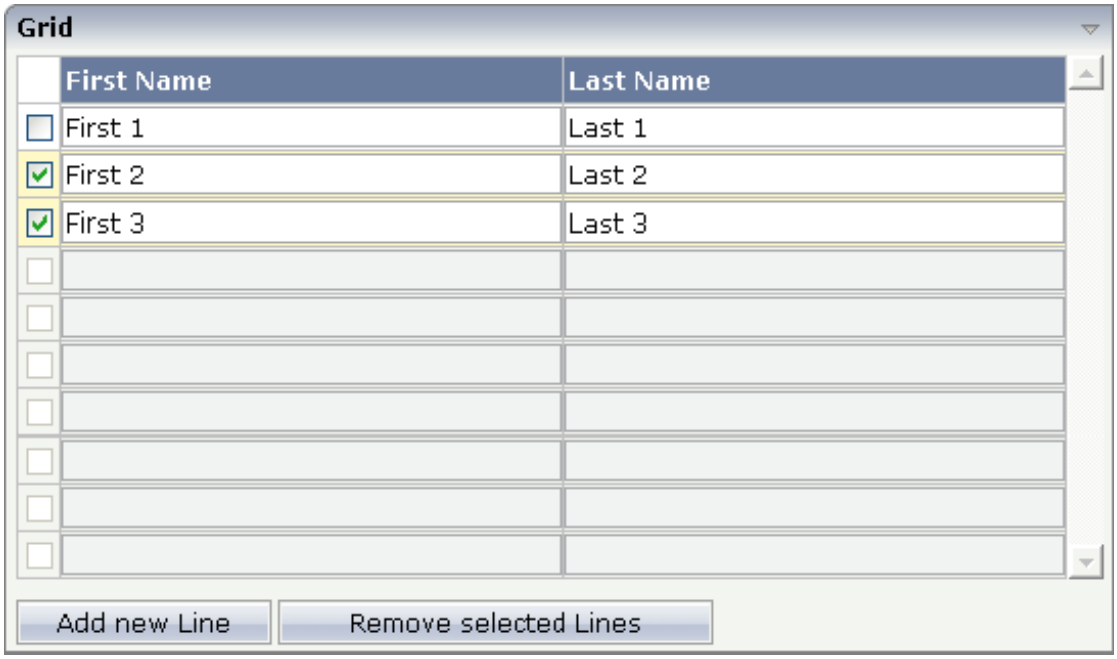
- Example for ROWTABLEAREA2 582
- Adapter Interface for ROWTABLEAREA2 584
- Built-in Events for ROWTABLEAREA2 584
- Making Grids Look like Grids 585
- Making Columns Movable 586
- Export to Clipboard and File 587
- Icon Bars 589
- ROWTABLEAREA2 Properties 596
- STR Properties 603

The ROWTABLEAREA2 is a container control that allows other controls to be arranged inside its grid management.

The ROWTABLEAREA2 control supports server-side scrolling and sorting. This concept is explained in *Server-Side Scrolling and Sorting*. An example for the usage of server-side scrolling and sorting with the ROWTABLEAREA2 control is contained in the Natural for Ajax demos.

Example for ROWTABLEAREA2

There is a grid that contains a header row and 10 lines. Each line contains one check box and two fields. Some of the lines are highlighted.



The XML layout definition is:

```
<rowarea name="Grid">
   <rowtablearea2 griddataprop="lines" rowcount="10" width="100%" withborder="true">
        <tr>
            <hdist>
            </hdist>
            <label name="First Name" asheadline="true">
            </label>
            <label name="Last Name" asheadline="true">
            </label>
        </tr>
        <repeat>
            <str valueprop="selected">
```

```
                <checkbox valueprop="selected" flush="screen" width="30">
                </checkbox>
                <field valueprop="firstname" width="50%">
                </field>
                <field valueprop="lastname" width="50%">
                </field>
            </str>
        </repeat>
    </rowtablearea2>
    <vdist height="10">
    </vdist>
    <itr>
        <button name="Add new Line" method="onAddLine">
        </button>
        <hdist>
        </hdist>
        <button name="Remove selected Lines" method="onRemoveLines">
        </button>
    </itr>
</rowarea>
```

Note the following:

- There is a ROWTABLEAREA2 definition with the property `griddataprop="lines"`. There is a `rowcount` definition of "10". This is the same as for the text grid processing: the grid container is bound to a server-side collection. Similar to the TEXTGRIDSSS2 definition, there is a row count that defines the number of lines.
- Inside the ROWTABLEAREA2 definition, there is first the definition of a normal table row (TR) in which a distance and two labels are defined. The labels are rendered with `asheadline="true"`.
- Inside the REPEAT definition, there is a special table row definition "STR" (selectable table row) that itself contains one CHECKBOX and two FIELD definitions. CHECKBOX and FIELDs are bound to properties themselves.
- After the ROWTABLEAREA2 definition, there is a vertical distance and a row that contains two buttons with which a user can manipulate the grid.

The content of the REPEAT block is repeated as many times as defined inside the `rowcount` definition of ROWTABLEAREA2. The content holds a table row (STR) - therefore the result is a grid.

Adapter Interface for ROWTABLEAREA2

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
```

If the grid has been configured for server-side scrolling and sorting, the data structure contains additional fields that control server-side scrolling and sorting (see below). In order to use server-side scrolling and sorting, set the property `natsss` in NATPAGE to "true".

```
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
1 LINESINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)
```

Built-in Events for ROWTABLEAREA2

Scrolling Events (Natural Server-Side Scrolling and Sorting only)

value-of-griddataprop.onTopindexChanged
value-of-griddataprop.onFirstPage (block scrolling only)
value-of-griddataprop.onLastPage (block scrolling only)
value-of-griddataprop.onPageDown (block scrolling only)
value-of-griddataprop.onPageUp (block scrolling only)

Selection Events (Natural Server-Side Scrolling and Sorting only)

value-of-griddataprop.onCtrlSelect
value-of-griddataprop.onSelect
value-of-griddataprop.onShiftSelect
value-of-griddataprop.onSelectAll
value-of-griddataprop.onDeselectAll

Sort Events (Natural Server-Side Scrolling and Sorting only)

value-of-griddataprop.onSort

Context Menu Events

value-of-griddataprop.reactOnContextMenuRequest

Making Grids Look like Grids

Fields typically contain a high number of FIELD controls. Typically, a FIELD control has a certain rendering that renders a field with a border and with a certain background color.

Be aware that inside the FIELD definition, there are two important properties:

- `noborder` - if set to "true", no border will be drawn
- `transparentbackground` - if set to "true", the field will always take over the background of the controls in which it is positioned (e.g. STR row).

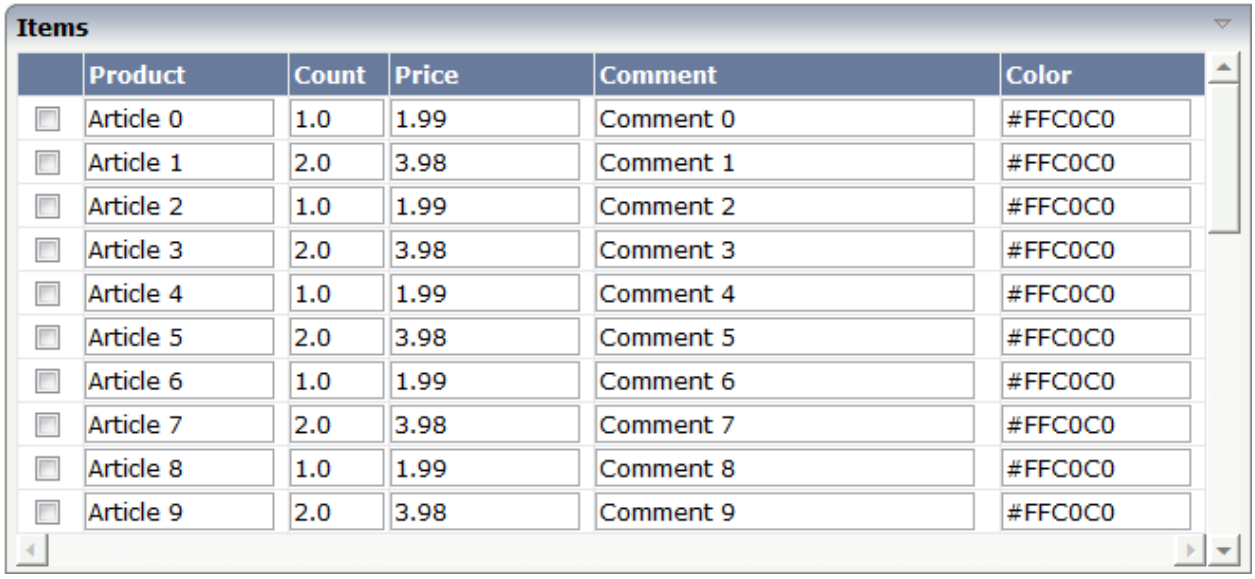
Have a look at the difference between the following screens. One screen uses the properties, the other screen does not use them.

This is a grid:

Items

	Product	Count	Price	Comment	Color
☐	Article 0	1.0	1.99	Comment 0	#FFC0C0
☐	Article 1	2.0	3.98	Comment 1	#FFC0C0
☐	Article 2	1.0	1.99	Comment 2	#FFC0C0
☐	Article 3	2.0	3.98	Comment 3	#FFC0C0
☐	Article 4	1.0	1.99	Comment 4	#FFC0C0
☐	Article 5	2.0	3.98	Comment 5	#FFC0C0
☐	Article 6	1.0	1.99	Comment 6	#FFC0C0
☐	Article 7	2.0	3.98	Comment 7	#FFC0C0
☐	Article 8	1.0	1.99	Comment 8	#FFC0C0
☐	Article 9	2.0	3.98	Comment 9	#FFC0C0

This is collection of fields:



Making Columns Movable

Columns in a ROWTABLEAREA2 control can be marked as movable so that these columns can be rearranged during execution time. The application program need not be changed.

To enable columns as movable columns, the following prerequisites must be fulfilled:

- The header row elements that are to represent movable columns must be defined with GRIDCOLHEADER controls.
- A specific column is defined as movable by setting the `movablecol` property of the GRIDCOLHEADER control to "true".
- The columns marked as movable must form a consecutive sequence. It is not possible to establish more than one sequence of movable columns within one grid.
- When columns are marked as movable, only FIELD and METHODLINK controls are allowed to be used in the table's body definition for the corresponding columns.

The following is an example of a layout definition with movable columns:

```
<rowarea name="Contents of the LINES structure">
    <rowtablearea2 griddataprop="lines" rowcount="15" width="100%"
     withborder="true">
        <tr>
            <label name="No." width="30" asheadline="true">
            </label>
            <label name="Dept" width="20%" asheadline="true">
            </label>
            <gridcolheader name="ID" width="20%" propref="id"
```

```
             withsorticon="false" movablecol="true">
            </gridcolheader>
            <gridcolheader name="Last Name" width="20%" propref="last"
             withsorticon="false" movablecol="true">
            </gridcolheader>
            <gridcolheader name="First Name" width="20%" propref="first"
             withsorticon="false" movablecol="true">
            </gridcolheader>
            <gridcolheader name="Reference" width="20%" propref="reflink"
             withsorticon="false" movablecol="true">
            </gridcolheader>
        </tr>
        <repeat>
            <str valueprop="selected">
                <selector valueprop="selected" singleselect="false">
                </selector>
                <treenode3 width="20%" withplusminus="true">
                </treenode3>
                <field valueprop="id" width="20%" noborder="true">
                </field>
                <field valueprop="last" width="20%" noborder="true">
                </field>
                <field valueprop="first" width="20%" noborder="true">
                </field>
                <methodlink method="reflinkclicked" valueprop="reflink"
                 width="20%">
                </methodlink>
            </str>
        </repeat>
    </rowtablearea2>
</rowarea>
```

The header row definition of the ROWTABLEAREA2 control contains a consecutive sequence of four GRIDCOLHEADER controls with the `movablecol` property set to "true". These GRIDCOLHEADER controls correspond to three FIELD controls and one METHODLINK control in the table's body definition.

Export to Clipboard and File

- Exported Columns
- Export of Grid Headers
- Exported Grid Items

- Enabling Grid Export

Exported Columns

The dynamic values of the controls in the grid columns are exported. Columns with the following controls are exported:

- field
- checkbox
- combodyn2
- dateinput
- text
- textout
- label
- methodlink
- radiobutton
- combofix

Additional static text, for example, a label with a static name property, is not exported. Columns which have been set to invisible via the GRIDCOLHEADER control are not exported.

Export of Grid Headers

Whether the grid headers are exported or not can be set dynamically at runtime. It can be set from within the Natural program, or the end-users can toggle the export dynamically via a corresponding icon. Headers which have been set to invisible via the GRIDCOLHEADER control are not exported.

Exported Grid Items

The following functions are supported.

- Copy all Grid Items to the Clipboard
- Copy the Selected Items to the Clipboard
- Export all Grid Items to a File

- Export the selected items to a file

Copy all Grid Items to the Clipboard

All items are copied to the clipboard. In some versions of Firefox, an additional confirmation pop-up opens, asking you to allow access to the clipboard. In Chrome and Internet Explorer 11/Edge this additional pop-up is not shown.

Copy the Selected Items to the Clipboard

The selected items are copied to clipboard. Users can select items via the normal multiselect functionality of ROWTABLEAREA2.

Export all Grid Items to a File

All items are written to a *.csv* (comma separated values) file. The browser's standard dialog for opening or saving a file opens.

Export the selected items to a file

The selected items are written to a *.csv* file. Users can select items via the normal multiselect functionality of ROWTABLEAREA2.

Enabling Grid Export

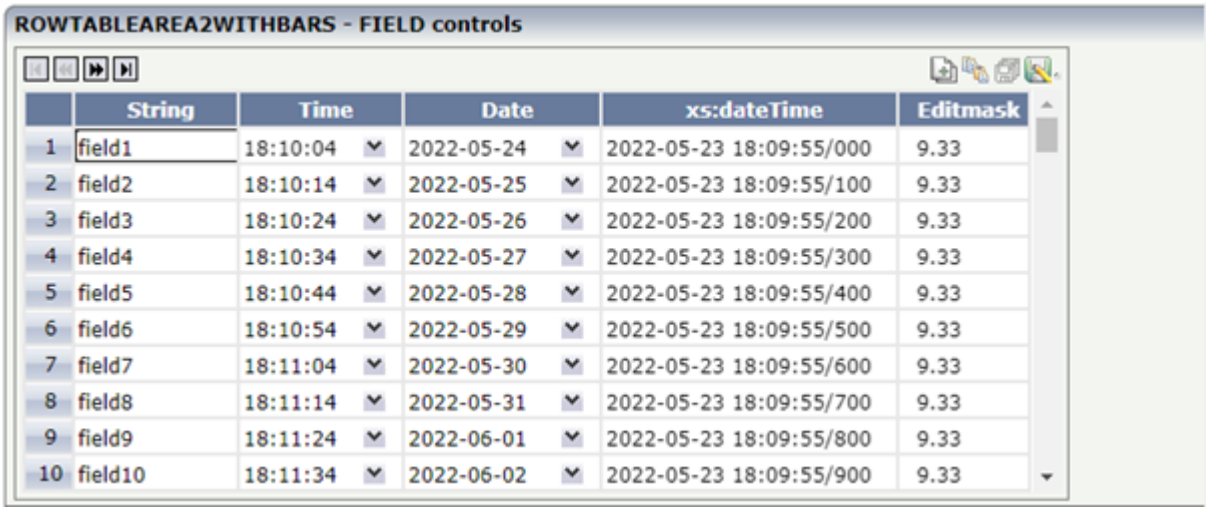
Either use the properties `withcopytoclipboard`, `withselectedtoclipboard`, `withexporttofile`, `withselectedtofile` and `withexportheaders` to add icons for these functions to your grid, or use the ROWTABLEAREA2WITHBARS control as explained in the chapter “Icon Bars” below. The ROWTABLEAREA2WITHBARS supports a more flexible way to arrange the icons.

Icon Bars

If you want to add multiple different functionalities like the export functions, block scrolling functions and other custom functions to your grid, you often want to arrange the different icons in bars. Use the ROWTABLEAREA2WITHBARS for this. Add an ICONBARLINE and a ROWTABLEAREA2 control as sub-nodes. You can add several icon bars in the ICONBARLINE control. BLOCKSCROLLINGBAR is an icon bar which contains the blockscrolling buttons. Use ICONBAR and ICONBARLINE controls to create your own icon bar. The icons TOCLIP-BOARDICON and TOFILEICON support the "copy to clipboard" and "export to file" functionalities. The EXPORTHEADERSTOGGLE supports the export of grid headers functionality.

Example:

```
<rowtablearea2withbars>
  <iconbarline>
    <iconbar float="right">
      <toclipboardicon>
      </toclipboardicon>
      <toclipboardicon selectedonly="true">
      </toclipboardicon>
      <tofileicon>
      </tofileicon>
      <tofileicon selectedonly="true">
      </tofileicon>
    </iconbar>
    <blockscrollingbar float="left">
    </blockscrollingbar>
  </iconbarline>
  <rowtablearea2 griddataprop="mygrid”
        …
  </rowtablearea2>
</rowtablearea2withbars>
```



	String	Time	Date	xs:dateTime	Editmask
1	field1	18:10:04	2022-05-24	2022-05-23 18:09:55/000	9.33
2	field2	18:10:14	2022-05-25	2022-05-23 18:09:55/100	9.33
3	field3	18:10:24	2022-05-26	2022-05-23 18:09:55/200	9.33
4	field4	18:10:34	2022-05-27	2022-05-23 18:09:55/300	9.33
5	field5	18:10:44	2022-05-28	2022-05-23 18:09:55/400	9.33
6	field6	18:10:54	2022-05-29	2022-05-23 18:09:55/500	9.33
7	field7	18:11:04	2022-05-30	2022-05-23 18:09:55/600	9.33
8	field8	18:11:14	2022-05-31	2022-05-23 18:09:55/700	9.33
9	field9	18:11:24	2022-06-01	2022-05-23 18:09:55/800	9.33
10	field10	18:11:34	2022-06-02	2022-05-23 18:09:55/900	9.33

The properties of the controls and icon bars mentioned above are shown in the tables below:

- Properties for ROWTABLEAREA2WITHBARS
- Properties for BLOCKSCROLLINGBAR
- Properties for ICONBAR
- Properties for ICONBARLINE
- Properties for TOCLIPBOARDICON
- Properties for TOFILEICON

- Properties for EXPORTHEADERSTOGGLE

Properties for ROWTABLEAREA2WITHBARS

Basic			
styleclass	CSS style class used for rendering.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%

Properties for BLOCKSCROLLINGBAR

Basic			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
styleclass	CSS style class used for rendering.	Optional	
float	If set to left, the bar is rendered on the left of the row. If set to right the bar is rendered at the right of the row. Default is right.	Optional	left right inherit

Properties for ICONBAR

Basic			
styleclass	CSS style class used for rendering.	Optional	
float	If set to left, the bar is rendered on the left of the row. If set to right the bar is rendered at the right of the row. Default is right.	Optional	left right inherit
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Properties for ICONBARLINE

Basic			
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Properties for TOCLIPBOARDICON

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	gif jpg jpeg
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	10 20 40 100 300
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	10 20 40 100 300
selectedonly	If set to true only the selected items of the grid are exported.	Optional	true false
format	If enabled, additional whitespace is inserted into each line so that column contents are vertically aligned when viewed in a standard text editor.	Optional	true false
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	

titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

Properties for TOFILEICON

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	gif jpg jpeg
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
imagewidth	Pixel width of the image that is shown inside the icon. If not defined then the icon is rendered with its normal width.	Optional	10 20 40 100 300
imageheight	Pixel height of the image that is shown inside the icon. If not defined then the icon is rendered with its normal height.	Optional	10 20 40 100 300
selectedonly	If set to true only the selected items of the grid are exported.	Optional	true false
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

Properties for EXPORTHEADERSTOGGLE

Basic			
valueprop	Name of the adapter parameter that represents the value of the control.	Obligatory	
trueimage	Image URL that is shown if the corresponding property value is "true".	Obligatory	gif jpg jpeg
falseimage	Image URL that is shown if the corresponding property value is "true".	Obligatory	gif jpg jpeg
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20").	Optional	

	(C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

ROWTABLEAREA2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
rowcount	Number of rows that are rendered inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property: If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that are defined as ROWCOUNT value. If a HEIGHT value is defined in addition (e.g. as percentage value "100%") then the number of rows depend on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that are picked from the server. You should define this value in a way so that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too	Optional	

	high ("100") because this causes more communication traffic and more rendering effort inside the browser.		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
firstrowcolwidths	If set to "true" then the grid is sized according to its first row. This first row typically is a header-TR-row in which GRIDCOLHEADER controls are used as column headers for the subsequent rows. Default is "false", i.e. the grid is sized according to its "whole content". Please note: when using the GRIDCOLHEADER control within the header-TR-row this property must be set to "true" - otherwise column resizing (by drag and drop) does not work correctly.	Sometimes obligatory	true false
onloadbehaviour	DEPRECATED: Loading behaviour of the items into the client. "block" (=default). Only "block" is supported´. All other settings are automatically mapped to "block".	Optional	block collection collectionorblock
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withborder	If set to "false" then no thin border is drawn around the controls that are contained in the grid. Default is "true".	Optional	true false
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "hidden".	Optional	auto scroll hidden
vscroll	Definition of the vertical scrollbar's appearance.	Optional	auto

	You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "scroll".		scroll hidden
firstrowcolwidths	(already explained above)		
frozencolumnsleft	Number of columns, which should be horizontally frozen to the left.	Optional	1 2 3 int-value
touchpadinput	If set to "true" then touch screen icons for scrolling are displayed in addition. Default is "false".	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%). Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.	Optional	1 2 3 int-value
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
darkbackground	Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used to integrate light coloured controls into darker container areas.	Optional	true false
invisiblemodeincompletelastrow	If set to "invisible" an incomplete last row is not shown.	Optional	invisible visible
withsliderfreeze	Setting this to "true" prevents unwanted slider jumps while scrolling up/down in a grid with a huge number of lines (for example 20000).	Optional	true false
Features			
clipboardaccess	If switched to true then the content of the grid can be selected and exported into the client's clipboard.	Optional	true false
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
withcopytoclipboard	If set to TRUE a corresponding icon to copy all grid items to the clipboard is added.	Optional	true false
copytoclipboardtitle	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	

formatcopytoclipboard	If enabled, additional whitespace is inserted into each line so that column contents are vertically aligned when viewed in a standard text editor.	Optional	true false
withselectedtoclipboard	If set to TRUE a corresponding icon to copy the selected grid items to the clipboard is added.	Optional	true false
selectedtoclipboardtitle	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
withexporttofile	If set to TRUE a corresponding icon to copy all grid items to a csv file is added.	Optional	true false
exporttofiletitle	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
withselectedtofile	If set to TRUE a corresponding icon to copy the selected grid items to a csv file is added.	Optional	true false
selectedtofiletitle	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
withexportheaders	If set to TRUE a corresponding icon for toggling the export of headers is added.	Optional	true false
exportheaderstitle	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	

exportheadersprop	Name of the adapter parameter that provides the value for exporting headers at runtime. At runtime "true" switches on the export of headers.	Optional	
Binding			
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in the grid, but not on an existing row, but in an empty area of the grid.	Optional	
fwdtabkeymethod	Name of the event that is sent to the adapter when the user presses the TAB key within the very last cell of the grid (last cell within the last line). Use property FWDTABKEYFILTER to associate this call with a grid column.	Optional	
fwdtabkeyfilter	By default the FWDTABKEYMETHOD is called if the user presses the TAB key within the veryfirst cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
bwdtabkeymethod	Name of the event that is sent to the adapter when the user presses SHIFT and TAB keys within the first cell of a grid line. Use property BWDTABKEYFILTER to associate this call with a cell of choice.	Optional	
bwdtabkeyfilter	By default the BWDTABKEYMETHOD is called if the user presses the SHIFT and TAB keys within the very first cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
frozencolumnsleftprop	Name of the adapter parameter which dynamically provides the number of columns to be be frozen to the left.	Optional	
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method	Optional	

	onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.		
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

STR Properties

STR (selectable table row) is a normal table row (TR) that highlights its background depending on an adapter property.

Basic			
valueprop	Name of the adapter parameter that defines if the row is selected or not.	Obligatory	
withalterbackground	Flag that indicates if the grid line shows alternating background color (like rows within a textgrids). Default is false. Please note: controls inside the row must have transparent background. In case of the FIELD control simply set property TRANSPARENTBACKGROUND to true.	Optional	true false
showifempty	Flag that indicates if an unused row is visible. Example: if set to false a grid with rowcount ten and a server side collection size of seven will hide the three remaining rows. Default is false.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	(already explained above)		
onclickmethod	Name of the event that is sent to the adapter when the user clicks.	Optional	
alwaysonclick	If no onclickmethod is defined, this property is ignored. If an onclickmethod is defined and this property is set to false, then the onclickmethod is only called if the clicked control does call a method itself. Example: A click on a button will only call the method defined in the button. Default is true, which means both methods will be called.	Optional	true false
ondblclickmethod	Name of the event that is sent to the adapter when the user double clicks.	Optional	
contextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in an empty area.	Optional	
proprefprop	Name of the adapter parameter that is filled when the user clicks a FIELD control. The VALUEPROP of the clicked field control will passed.	Optional	
strstyleprop	Name of the adapter parameter that dynamically provides explicit style information for the control.	Optional	
backgroundcolorprop	Name of the adapter parameter that dynamically provides the background color for this control.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group	Optional	

	structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.		
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

78 ROWTABLEAREA3 - The Array Grid

- Example for ROWTABLEAREA3 608
- Adapter Interface for ROWTABLEAREA3 610
- Built-in Events for ROWTABLEAREA3 613
- ROWTABLEAREA3 Properties 613
- TR3 Properties 619
- GRIDCOLHEADER3 Properties 620
- STR3 Properties 622
- FIELD3 Properties 623

The ROWTABLEAREA3 control is a specialized ROWTABLEAREA2 grid which is used to bind a two-dimensional Natural array to a grid. The grid cells are rendered as FIELD controls.

Other than with the ROWTABLEAREA2 control, you do not need to define the number of grid columns at design time. Instead, you define the number of columns and rows dynamically at runtime.

The ROWTABLEAREA3 control supports server-side scrolling and sorting, just like the ROWTABLEAREA2 control. This concept is explained in *Server-Side Scrolling and Sorting*. An example for the usage of server-side scrolling and sorting with the ROWTABLEAREA2 control is contained in the Natural for Ajax demos.

Example for ROWTABLEAREA3

You can find the following example in the Natural for Ajax demos.

Two-Dimensional Demo Array

Days	San Francisco	New York	Paris	London	Darmstadt
1	Sunny	Snowy	Cloudy	Sunny	Sunny
2	Sunny	Snowy	Windy		Sunny
3	Foggy	Stormy	Rainy	Foggy	Sunny
4		Stormy	Windy	Cloudy	Windy
5	Sunny	Sunny	Stormy	Stormy	Sunny
6	Sunny	Sunny	Sunny	Foggy	Sunny
7	Sunny	Sunny		Rainy	Sunny

The XML layout definition for the grid is:

```
<rowarea name="Two-Dimensional Demo Array">
    <rowtablearea3 griddataprop="gridArray" gridpropsprop="gridArrayProps"
     rowcount="7" firstrowcolwidths="true" width="100%">
       <tr3>
         <gridcolheader textalign="center" name="Days" width="60"
          propref="selected" withsorticon="false">
         </gridcolheader>
         <gridcolheader3 nameprop="gridColHeaderDays"
          widthprop="headerDaysWidthProp">
         </gridcolheader3>
         <hdist width="100%"></hdist>
       </tr3>
       <repeat3>
         <str3 valueprop="selected" withalterbackground="true">
```

```
          <selector valueprop="selected" width="60">
          </selector>
          <field3 width="100%" statusprop="gridStatus" titleprop="gridTitle"
           noborder="true" datatype="string">
          </field3>
          <hdist width="100%"></hdist>
        </str3>
      </repeat3>
    </rowtablearea3>
</rowarea>
```

The above layout contains three major parts that you need to define for your grid. These parts are described in the topics below.

- Defining the General Grid Rendering and Data Binding
- Defining the Grid Headers
- Defining the Grid Content

Defining the General Grid Rendering and Data Binding

In the ROWTABLEAREA3 control, the properties `griddataprop` and `gridpropsprop` define the bindings to the Natural application. Other than the ROWTABLEARE2 control, the ROWTABLEAREA3 control supports separate properties for the data shown in the grid cells (`griddataprop`) and the rendering data for the grid cells (`gridpropsprop`). Other properties such as `rowcount`, `firstrowcolwidth` and `width` are used in the same way as in the ROWTABLEAREA2 control.

Defining the Grid Headers

The TR3 control is a specialized TR container control that is used to define the grid headers. Since the column count is dynamically defined at runtime by the Natural application, you simply define one GRIDCOLHEADER3 control to render all headers of the grid data.

The GRIDCOLHEADER3 control is a specialized GRIDCOLHEADER control. Other than with the GRIDCOLHEADER control, width and name values are dynamically specified at runtime. For this purpose, the GRIDCOLHEADER3 control supports properties such as `nameprop` and `widthprop`.

In the above example, a selector column is defined as the first column, in addition to the real data columns. This selector column does not contain business data; its purpose is just rendering. Therefore, the header for the selector is defined via a GRIDCOLHEADER control and the corresponding rendering values are defined at design time.

GRIDCOLHEADER3 refers to the two-dimensional Natural data array. GRIDCOLHEADER refers to the specific SELECTOR column.

Defining the Grid Content

The grid content is defined within a REPEAT3 block. The REPEAT3 control is a specialized REPEAT control. As in ROWTABLEAREA2, its content defines a row which is repeated within the grid as defined by the `rowcount` property. Besides the real data shown in a grid, a row can have an additional SELECTOR control and can make use of the HDIST control to fine-tune the rendering in the browser.

The FIELD3 control is a specialized FIELD control which defines the rendering of the grid cells. Since the number of cells is dynamically defined at runtime, you just define one FIELD3 control to specify the rendering of all grid cells in a homogenous way.

The STR3 control is a specialized STR control which contains the above-mentioned parts of a row: SELECTOR (optional), FIELD3 and HDIST (optional).

Adapter Interface for ROWTABLEAREA3

In the parameter data area of the adapter, the grid data of the above example is represented by the data structures described below.

Two-Dimensional Array for the Grid Data

```
1 GRIDARRAY (A/1:*,1:*) DYNAMIC
```

The name of the data structure is defined in the `griddataprop` property of the ROWTABLEAREA3 control. The specific type of a single cell - here `(A) DYNAMIC` - can be defined in the `datatype` property of the FIELD3 control.

Data Structure for Rendering

```
1 GRIDARRAYPROPS
2 HEADERS (1:*)
3 GRIDCOLHEADERDAYS (A) DYNAMIC
3 HEADERDAYSWIDTHPROP (A) DYNAMIC
2 ROWS (1:*)
3 COLS (1:*)
4 GRIDSTATUS (A) DYNAMIC
4 GRIDTITLE (A) DYNAMIC
3 SELECTED (L)
```

By default, the name of the data structure is the same as the name of the two-dimensional array (see above), followed by "PROPS". Alternatively, you can specify your own name in the `gridpropsprop` property of the ROWTABLEAREA3 control.

This data structure contains rendering information for the headers and all grid cells. `HEADERS`, `ROWS` and `COLS` are predefined names and are added to this structure correspondingly. Other fields such

as GRIDCOLHEADERDAYS, HEADERDAYSWIDTHPROP, GRIDSTATUS, GRIDTITLE and SELECTED are generated based on the properties specified in GRIDCOLHEADER3, FIELD3, HDIST and the container controls of the grid.

Data Structure for Server-Side Scrolling and Sorting (Optional)

```
1 GRIDARRAYINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (A) DYNAMIC
2 TOPINDEX (I4)
```

If the grid has been configured for server-side scrolling and sorting, the data structure contains additional fields that control server-side scrolling and sorting. The usage is the same as in the ROWTABLEAREA2 control.

By default, the name of the data structure is the same as the name of the two-dimensional array (see above), followed by "INFO". Alternatively, you can specify your own name in the gridinfoprop property of the ROWTABLEAREA3 control.

Filling the Data Structure at Runtime

The advantage of the ROWTABLEAREA3 grid is that it separates the data structure for the business data (the GRIDARRAY above) from the data structure for the rendering data (the GRIDARRAYPROPS above). Having two separate data structures allows you to directly bind grids to existing business data. However, having two data structures requires making the two data structures the same size in regard to rows and columns. The Natural application is in charge of resizing the corresponding data structures. See the examples in the Natural for Ajax demos for a simple implementation pattern.

Using Natural Control Variables with Array Grids

To easily find out which cells of the grids have been modified, you can specify the name of a Natural control variable in the njx:natcv property of the ROWTABLEAREA3 control. The control variable can be defined within the grid data structure or outside of the grid data structure.

Example: Control Variable within the Grid Data Structure

```
<rowtablearea3 griddataprop="lines" njx:natcv="lines(*,*).mycell-cv" rowcount="5" >
   <repeat3>
      <str3 valueprop="selected">
         <selector valueprop="selected">
         </selector>
         <field3 titleprop="fld3titleprop" noborder="true" ↩
transparentbackground="true">
         </field3>
         <hdist width="100%">
         </hdist>
         <xcidatadef dataprop="mycell-cv" datatype="C">
         </xcidatadef>
       </str3>
    </repeat3>
</rowtablearea3>
```

This will generate the following Natural data structure:

```
1 LINESPROPS
2 ROWS (1:*)
3 COLS (1:*)
4 FLD3TITLEPROP (A) DYNAMIC
4 MYCELL-CV (C)
3 SELECTED (L)
1 LINES (A/1:*,1:*) DYNAMIC
```

Example: Control Variable outside of the Grid Data Structure

```
<rowtablearea3 griddataprop="lines" njx:natcv="mycell-cv(*,*)"  rowcount="5" >
  <repeat3>
     <str3 valueprop="selected">
        <selector valueprop="selected">
        </selector>
        <field3 titleprop="fld3titleprop" noborder="true" ↩
transparentbackground="true">
        </field3>
        <hdist width="100%">
        </hdist>
      </str3>
  </repeat3>
</rowtablearea3>
...
<xcidatadef dataprop="mycell-cv" datatype="C" array="true" arraydimension="2">
</xcidatadef>
```

This will generate the following Natural data structure:

```
1 LINESPROPS
2 ROWS (1:*)
3 COLS (1:*)
4 FLD3TITLEPROP (A) DYNAMIC
3 SELECTED (L)
1 LINES (A/1:*,1:*) DYNAMIC
1 MYCELL-CV (C/1:*,1:*)
```

Built-in Events for ROWTABLEAREA3

Scrolling Events (Natural Server-Side Scrolling and Sorting only)

value-of-griddataprop.onTopindexChanged
value-of-griddataprop.onFirstPage (block scrolling only)
value-of-griddataprop.onLastPage (block scrolling only)
value-of-griddataprop.onPageDown (block scrolling only)
value-of-griddataprop.onPageUp (block scrolling only)

Selection Events (Natural Server-Side Scrolling and Sorting only)

value-of-griddataprop.onCtrlSelect
value-of-griddataprop.onSelect
value-of-griddataprop.onShiftSelect
value-of-griddataprop.onSelectAll
value-of-griddataprop.onDeselectAll

Sort Events (Natural Server-Side Scrolling and Sorting only)

value-of-griddataprop.onSort

Context Menu Events

value-of-griddataprop.reactOnContextMenuRequest

ROWTABLEAREA3 Properties

Basic			
rowcount	Number of rows that are rendered inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property:	Optional	1 2 3 int-value

	If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that are defined as ROWCOUNT value. If a HEIGHT value is defined in addition (e.g. as percentage value "100%") then the number of rows depend on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that are picked from the server. You should define this value in a way so that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too high ("100") because this causes more communication traffic and more rendering effort inside the browser.		
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
width	Width of the control. There are three possibilities to define the width:	Sometimes obligatory	100 120 140

	(A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		160 180 200 50% 100%
firstrowcolwidths	If set to "true" then the grid is sized according to its first row. This first row typically is a header-TR-row in which GRIDCOLHEADER controls are used as column headers for the subsequent rows. Default is "false", i.e. the grid is sized according to its "whole content". Please note: when using the GRIDCOLHEADER control within the header-TR-row this property must be set to "true" - otherwise column resizing (by drag and drop) does not work correctly.	Sometimes obligatory	true false
maxcols	Per default the grid can contain a maximum of 50 columns. Set this property if you want to increase or decrease the default for performance reasons.	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			

withborder	If set to "false" then no thin border is drawn around the controls that are contained in the grid. Default is "true".	Optional	true false
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "hidden".	Optional	auto scroll hidden
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "scroll".	Optional	auto scroll hidden
firstrowcolwidths	(already explained above)		
clipboardaccess	If switched to true then the content of the grid can be selected and exported into the client's clipboard.	Optional	true false
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
touchpadinput	If set to "true" then touch screen icons for scrolling are displayed in addition. Default is "false".	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%).	Optional	1 2 3

	Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.		int-value
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
darkbackground	Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used. If DARKBACKGROUND is set to true then the darker background colour is chosen. This property typically is used to integrate light coloured controls into darker container areas.	Optional	true false
invisiblemodeincompletelastrow	If set to "invisible" an incomplete last row is not shown.	Optional	invisible visible
Binding			
fwdtabkeyfilter	By default the FWDTABKEYMETHOD is called if the user presses the TAB key within the veryfirst cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	

bwdtabkeyfilter	By default the BWDTABKEYMETHOD is called if the user presses the SHIFT and TAB keys within the very first cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	

njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

TR3 Properties

Basic			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). Please note: the row content may overrule this setting. The height setting "100px" of an embedded textbox will beat a row height of "50px". (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
withalterbackground	Flag that indicates if the grid line shows alternating background color (like rows within a textgrids). Default is false. Please note: controls inside the row must have transparent background. In case of the FIELD control simply set property TRANSPARENTBACKGROUND to true.	Optional	true false
trstyle	CSS style definition that is directly passed into this control.	Optional	background-color: #FF0000 color: #0000FF

	With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		font-weight: bold
trstyleprop	Name of the adapter parameter that dynamically provides explicit style information for the control.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

GRIDCOLHEADER3 Properties

Basic			
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Sometimes obligatory	
textidprop	Name of the adapter property that provides the multi language dependent text which is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
widthprop	Name of the adapter parameter that provided the width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this	Obligatory	

	control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
Appearance			
nowrap	The textual content of the header is not wrapped automatically. No line break will be performed automatically by the browser. If you want the text of the header to be wrapped, set the value to "false".	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2 VAR3 VAR4
textalign	Alignment of text inside the control.	Optional	left center right
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It	Optional	1 2 3 4 5 50

	does not make sense in ITR rows, because these rows are explicitly not synched.		int-value
Binding			
nameprop	(already explained above)		
textidprop	(already explained above)		
widthprop	(already explained above)		
titleprop	Name of the adapter parameter that provides the text which is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextidprop	Name of the adapter parameter that provides the text ID which is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
withsorticonprop	Name of the adapter parameter that provides the flag which indicates if a small sort indicator is shown within the right corner of the control. Default is TRUE.	Optional	
sorttitleprop	Name of the adapter parameter that provides the text which is shown as tooltip for the sort indicator. Either input text by using this SORTTITLE property - or use the SORTTITLETEXTID in order to define a language dependent literal.	Optional	
sorttitletextidprop	Name of the adapter parameter that provides the text ID which is passed to the multi language management - representing the tooltip text for the sort indicator.	Optional	
Comment			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

STR3 Properties

Basic			
withalterbackground	Flag that indicates if the grid line shows alternating background color (like rows within a textgrids). Default is false. Please note: controls inside the row must have transparent background. In case of the FIELD control simply set property TRANSPARENTBACKGROUND to true.	Optional	true false
showifempty	Flag that indicates if an unused row is visible. Example: if set to false a grid with rowcount ten and a server side collection size of seven will hide the three remaining rows.	Optional	true false

	Default is false.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

FIELD3 Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	(already explained above)		
length	Width of FIELD in amount of characters. WIDTH and LENGTH should not be used together. Note that the actual size of the control depends on the font definition if using the LENGTH property.	Optional	5 10 15 20 int-value
maxlength	Maximum number of characters that a user may enter. This property is not depending on the LENGTH property - please do not get confused by the similar naming. MAXLENGTH has nothing to do with the optical sizing of the control but only with the number of characters you may input.	Optional	5 10 15 20 int-value
autotab	If set to true, an automatic tab is executed for fields with a specified MAXLENGTH when the maxlength value is reached. For fields without a MAXLENGTH specified it has no effect. Default is true.	Optional	true false
textalign	Alignment of text inside the control.	Optional	left center right
password	If set to "true", each entered character is displayed as a '*'.	Optional	true false
displayonly	If set to true, the FIELD will not be accessible for input. It is just used as an output field.	Optional	true false
direction	Presets the default(BiDi) direction of the control. Use black string in order to have the default value.	Optional	rtl ltr

uppercase	If "true" then all input is automatically transferred to upper case characters.	Optional	true false
align	Horizontal alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control itself. In this case the "align" property specifies the position of the control inside the column. In most cases you do not require the align control to be explicitly defined because the size of the column around the controls exactly is sized in the same way as the contained control. If you want to directly control the alignment of text: in most text based controls there is an explicit property "textalign" in which you align the control's contained text.	Optional	left center right
valign	Vertical alignment of control in its column. Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimes the size of the column is bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.	Optional	top middle bottom
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3 4

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		5 50 int-value
fieldstyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
noborder	Boolean value defining if the control has a border. Default is "false".	Optional	true false
transparentbackground	Boolean value defining if the control is rendered with a transparent background. Default is "false".	Optional	true false
invisiblemode	If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false": (1) "invisible": the control is not visible. (2)"cleared": the control is not visible but it still occupies space.	Optional	invisible cleared
Binding			
alwaysflush	If set to TRUE then a specified server flushmethod is also called in case the value has not changed. The default is FALSE, meaning that a server flushmethod is only called for a changed value.	Optional	true false
flush	Flushing behaviour of the input control.	Optional	screen

	By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user e.g. presses a button. By using the FLUSH property you can change this behaviour. Setting FLUSH to "server" means that directly after changing the input a synchronization with the server adapter is triggered. As consequence you directly can react inside your adapter logic onto the change of the corresponding value. - Please be aware of that during the synchronization always all changed properties - also the ones that were changed before - are transferred to the adapter object, not only the one that triggered the synchronization. Setting FLUSH to "screen" means that the changed value is populated inside the page. You use this option if you have redundant usage of the same property inside one page and if you want to pass one changed value to all its representation directly after changing the value.		server
textidmode	If using property "valuetextprop" then a field knows an id and a text for a certain value. There are three types of display: either both are shown together, separated by an "-" (e.g. "id - text"). Or only text is shown or only the id is shown. If not defined at all then the system's default text id-mode will be chosen. The default mode can be defined as part of the CIS session context.	Optional	0 1 2
Validation			
datatype	By default, the FIELD control is managing its content as string. By explicitly setting a datatype you can define that the control... ...will check the user input if it reflects the datatype. E.g. if the user inputs "abc" into a field with datatype "int" then a corresponding error message will popup when the user leaves the field. ...will format the data coming from the server or coming form the user input: if the field has datatype "date" and the user inputs "010304" then the input will be translated into "01.03.2004" (or other representation, dependent on date format settings).	Optional	date float int long time timestamp color xs:decimal xs:double xs:date

	In addition value popups are offered for the user automatically for some datatypes: e.g. when specifying datatype "date" the automatically the field provides a calendar input popup. Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.		xs:dateTime xs:time
editmask	NATPAGE only: A subset of the Natural edit masks is supported depending on the data type.	Optional	
validationrules	Contains information used for Data Validation. Use the Validation Rules Editor to make changes!	Optional	
validation	Regular expression against which the content of the field is checked on client side when the user changes the field. If the validation fails then an error message popup up and informs the user about the wrong input.	Optional	[a-zA-Z0-9_.-] {1,}\ \@[a-zA-Z0-9_.-] {1,}\ \.\ \w{2,}\ \d{5} [0-9)(-/+]+
validationuserhint	If a client side validation fails due to wrong user input then an error popup is opened. If you define a hint inside this property then the hint is output to the user in order to tell in which way to input the value. The hint is not language dependent.	Optional	
digits	Number that specifiies how many digits are to be displayed (i.e. digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.	Optional	1 2 3 int-value
decimaldigits	Specifies the number of displayed decimal digits. If using this feature then the DATATYPE property must be set to 'float'.	Optional	1 2 3 int-value
spinrangemin	An integer value which defines the lower bound of the value range.	Optional	1 2

			3 int-value
spinrangemax	An integer value which defines the upper bound of the value range.	Optional	1 2 3 int-value
Valuehelp			
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the user can input a value either directly via keyboard or by using the popupmethod's help. If set to "true" then no keyboard input is possible - but only selection from the popup-method's help.	Optional	true false
popuponalt40	Value help in a field is triggered either by clicking with the mouse or by pressing a certain key inside the field. The "traditional" keys are "cursor-down", "F7" or "F4". Sometimes you do not want to mix other "cursor-down" behaviour (e.g. scrolling in lists) with the value help behaviour. In this case switch this property to "true" - and the value help will only come up anymore when "alt-cursor-down" is pressed.	Optional	true false
popupcombowidth	Pixel width of the standard "openIdValueCombo" popup dialog. Default is field width or at least 150 pixel.	Optional	1 2 3 int-value
popupicon	URL of image that is displayed inside the right corner of the field to indicate to the user that there is some value help available.. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL: (A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying	Optional	gif jpg jpeg

	"../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
touchpadinput	Boolean property that decides if touch pad support is offered for the FIELD control. The default is "false". If switched to "true" then you can input data into the field via a touch pad. As consequence you can use this control for making inputs through a touch terminal.	Optional	true false
onlinehelp			
helpid	Help id that is passed to the online help management in case the user presses F1 on the control.	Optional	
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
formula	Contains information used by the Formula Editor. Use the Formula Editor to make changes!	Optional	
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

autocallpopupmethodoffset	The offset (milliseconds) after the last key down event for calling the AUTOCALLPOPUPMETHOD. Makes only sense if an AUTOCALLPOPUPMETHOD is specified.	Optional	1 2 3 int-value

79 FLEXLINE - Flexible Columns in Control Grids

- Example 634
- Adapter Interface 636
- ATTRIBUTES 636
- FLEXLINE Properties 636

The FLEXLINE control offers the option to define the columns of a grid dynamically at runtime. That is: the application decides at runtime which column controls to use with which properties.

Example

The following example shows a ROWTABLEAREA2 control containing a FLEXLINE control:



The XML layout definition is:

```
<rowarea name="Move Columns">
  <rowtablearea2 griddataprop="lines" rowcount="5"
  width="100%" firstrowcolwidths="true">
    <tr>
      <label width="30" asplaintext="true"></label>
      <flexline infoprop="headerFL"></flexline>
      <hdist width="100%"></hdist>
    </tr>
    <repeat>
      <str valueprop="lineSelected">
        <selector valueprop="lineSelected" width="30"></selector>
        <flexline infoprop="/contentFL"></flexline>
        <hdist width="100%"></hdist>
        <xcidatadef dataprop="nr" clientdata="true"></xcidatadef>
        <xcidatadef dataprop="description" clientdata="true"></xcidatadef>
        <xcidatadef dataprop="price" clientdata="true"></xcidatadef>
        <xcidatadef dataprop="quantity" clientdata="true"></xcidatadef>
      </str>
    </repeat>
  </rowtablearea2>
</rowarea>
```

The grid uses two FLEXLINE controls: One for the grid headers and one for the grid lines. The controls to be used for the columns in a line are completely defined dynamically at runtime. The FLEXLINE control for the grid headers is dynamically filled with GRIDCOLHEADER controls at runtime. Therefore, all features of the GRIDCOLHEADER control (such as moving and resizing columns) are available in the grid.

The FLEXLINE control "headerFL", which is used for the headline of the table, binds to the following Natural data structure:

```
1 HEADERFL (1:*)
2 ATTRIBUTES (A) DYNAMIC
2 CONTROL (A) DYNAMIC
2 VALUEPROP (A) DYNAMIC
2 WIDTH (A) DYNAMIC
```

The FLEXLINE control "/contentFL" (note the initial slash which must be used for controls inside a grid), which is used for the lines of the table, binds to the following Natural data structure:

```
1 CONTENTFL (1:*)
2 ATTRIBUTES (A) DYNAMIC
2 CONTROL (A) DYNAMIC
2 VALUEPROP (A) DYNAMIC
2 WIDTH (A) DYNAMIC
```

As `CONTROL` value, the name of the control is passed. Valid values are:

BUTTON
CHECKBOX
COMBODYN2
DATEINPUT
FIELD
GRIDCOLHEADER
ICON
IMAGEOUT
METHODLINK
TEXT
TEXTOUT
TOGGLE

The `WIDTH` value must be set for all controls. The allowed values depend on the specific control.

If the control has a `valueprop` property, the field `VALUEPROP` must contain the name of the `valueprop` for this control. For GRIDCOLHEADER, the `VALUEPROP` must contain the value of the `propref` property.

`ATTRIBUTES` contains additional control properties that you want to use for the control as a semi-colon-separated list of name-value pairs. Instead of defining the properties for a control at design time in the Layout Painter, you now pass them dynamically at runtime. Example for the FIELD control: "noborder;true".

The `VALUEPROP` value defined in the `CONTENTFL` data structure refers to elements of the `LINES` data structure. In the following example, the property "nr" is defined as element of the `LINES` data structure using the XCIDATADEF control.

```
EDITLINE.CONTROL(1):="field"
EDITLINE.VALUEPROP(1):="nr"
EDITLINE.WIDTH(1):="100%"
COMPRESS "noborder;true;transparentbackground;true" INTO CONTENTFL.ATTRIBUTES(1)" ↩
LEAVING NO SPACE
```

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
1 CONTENTFL (1:*)
2 ATTRIBUTES (A) DYNAMIC
2 CONTROL (A) DYNAMIC
2 VALUEPROP (A) DYNAMIC
2 WIDTH (A) DYNAMIC
```

ATTRIBUTES

As you can see in the above example, the properties of the controls are passed as a comma-separated string value of the `ATTRIBUTES` data field. If one of the property values contains a semicolon (;) itself, you must use a backslash followed by a semicolon (\;) in the property value. Example for edit mask properties in a FIELD control: "datatype;P5.2\;999.99".

You can use and combine any properties that are available for the controls with the exception of the `njx:*` properties (such as `njx:natname` or `njx:natcomment`). The `njx:*` properties are used to adapt the code generation of the corresponding Natural adapter code. Since no Natural adapter code is generated at runtime, these properties cannot be used in the `ATTRIBUTES` data field of the FLEXLINE control.

FLEXLINE Properties

Basic			
infoprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
withborder	Flag that indicates if a border is drawn between the controls that are rendered inside the FLEXLINE control. Default is "false", i.e. no border is drawn.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

80 MGDGRID - Managing the Grid

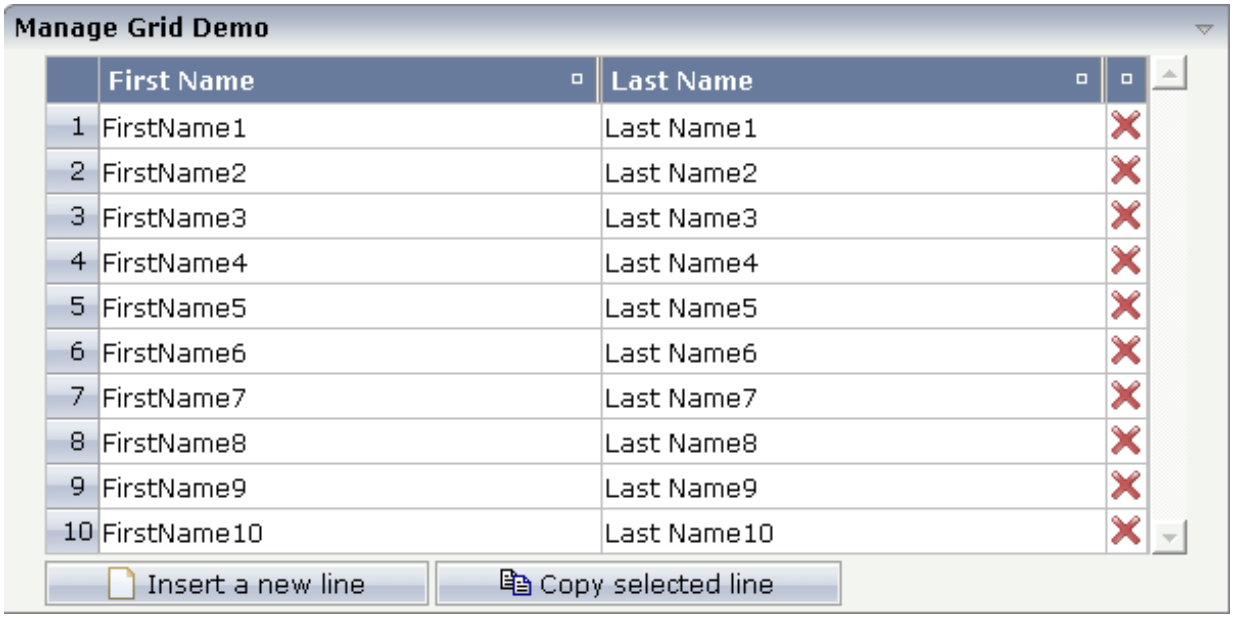
- Example 641
- Adapter Interface 642
- Built-in Events 643
- MGDGRID Properties 643
- ROWINSERT Properties 648
- ROWCOPY Properties 648
- ROWDELETE Properties 649

The MGDGRID control is an extension of the **ROWTABLEAREA2** control. It allows to insert, copy and delete rows of the grid.

Like the ROWTABLEAREA2 control, the MGDGRID control supports server-side scrolling and sorting. This concept is explained in *Server-Side Scrolling and Sorting*. An example for the usage of server-side scrolling and sorting with the ROWTABLEAREA2 control is contained in the Natural for Ajax demos. The same example can be used to illustrate the usage of server-side scrolling and sorting with the MGDGRID control.

See also ***STR Properties*** which are described with the ROWTABLEAREA2 control.

Example



There is a grid that contains a header row and 10 lines. Each line contains two fields and a “delete row” control.

Each of the function controls (insert, copy, delete) can be added at the top of the MGDGRID, below the MGDGRID or within the lines of the MGDGRID.

Look at the corresponding layout definition:

```
<rowarea name="Manage Grid Demo">
 <mgdgrid griddataprop="mglines" rowcount="10" width="100%" firstrowcolwidths="true">
   <tr>
     <label name="  " width="25" asheadline="true">
     </label>
     <gridcolheader name="First Name" width="50%">
     </gridcolheader>
     <gridcolheader name="Last Name" width="50%" >
     </gridcolheader>
     <gridcolheader width="20">
     </gridcolheader>
     <hdist></hdist>
   </tr>
   <repeat>
     <str valueprop="selected" showifempty="true">
       <selector valueprop="selected" singleselect="true">
       </selector>
       <field valueprop="fname" width="100%">
```

```
        </field>
        <field valueprop="lname" width="100%">
        </field>
        <rowdelete>
        </rowdelete>
      </str>
    </repeat>
    <mgdfunctions>
      <rowinsert title="Insert a new line">
      </rowinsert>
      <rowcopy title="Copy selected line">
      </rowcopy>
    </mgdfunctions>
  </mgdgrid>
</rowarea>
```

The MGDGRID control is an extension to the ROWTABLEAREA2 control. See the description of the **ROWTABLEAREA2** control for further information.

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
1 MGLINES (1:*)
2 FNAME (U) DYNAMIC
2 LNAME (U) DYNAMIC
2 SELECTED (L)
```

If the grid has been configured for server-side scrolling and sorting, the data structure contains additional fields that control server-side scrolling and sorting (see below). In order to use server-side scrolling and sorting, set the property `natsss` in NATPAGE to "true".

```
1 MGLINES (1:*)
2 FNAME (U) DYNAMIC
2 LNAME (U) DYNAMIC
2 SELECTED (L)
1 LINESINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4) ↩
```

Built-in Events

value-of-griddataprop.onCtrlSelect
value-of-griddataprop.onSelect
value-of-griddataprop.onShiftSelect
value-of-griddataprop.onSort
value-of-griddataprop.onTopindexChanged

MGDGRID Properties

Basic			
griddataprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
rowcount	Number of rows that are rendered inside the control. There are two ways of using this property - dependent on whether you in addition define the HEIGHT property: If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that are defined as ROWCOUNT value. If a HEIGHT value is defined in addition (e.g. as percentage value "100%") then the number of rows depend on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that are picked from the server. You should define this value in a way so that it is not too low - otherwise your grid will not be fully filled. On the other hand it should not be defined too high ("100") because this causes more communication traffic and more rendering effort inside the browser.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content.	Optional	100 150 200 250 300

	(B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		250 400 50% 100%
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Sometimes obligatory	100 120 140 160 180 200 50% 100%
firstrowcolwidths	If set to "true" then the grid is sized according to its first row. This first row typically is a header-TR-row in which GRIDCOLHEADER controls are used as column headers for the subsequent rows. Default is "false", i.e. the grid is sized according to its "whole content". Please note: when using the GRIDCOLHEADER control within the header-TR-row this property must be set to "true" - otherwise column resizing (by drag and drop) does not work correctly.	Sometimes obligatory	true false

comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withborder	If set to "false" then no thin border is drawn around the controls that are contained in the grid. Default is "true".	Optional	true false
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "hidden".	Optional	auto scroll hidden
vscroll	Definition of the vertical scrollbar's appearance. You can define that scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "scroll".	Optional	auto scroll hidden
firstrowcolwidths	(already explained above)		
clipboardaccess	If switched to true then the content of the grid can be selected and exported into the client's clipboard.	Optional	true false
withblockscrolling	If switched to "true" then the grid will show small scroll icons by which the user can scroll the grid's content. Scrolling typically is done by using the grid's scrollbar - the scroll icons that are switched on by this property are an additional possibility to scroll.	Optional	true false
touchpadinput	If set to "true" then touch screen icons for scrolling are displayed in addition. Default is "false".	Optional	true false
requiredheight	Minimum height of the control in pixels. Use this property to ensure a minimum height if the overall control's height is a percentage of the available space - i.e. if value of property HEIGHT is a percentage (e.g. 100%).	Optional	1 2 3 int-value

	Please note:You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less than the required height the end of the control is just cut off.		
tablestyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon. Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in the grid, but not on an existing row, but in an empty area of the grid.	Optional	
fwdtabkeymethod	Name of the event that is sent to the adapter when the user presses the TAB key within the very last cell of the grid (last cell within the last line). Use property FWDTABKEYFILTER to associate this call with a grid column.	Optional	
fwdtabkeyfilter	By default the FWDTABKEYMETHOD is called if the user presses the TAB key within the veryfirst cell of the grid. Input the name of a cell's VALUEPROP to associate the method call with any other column.	Optional	
bwdtabkeymethod	Name of the event that is sent to the adapter when the user presses SHIFT and TAB keys within the first cell of a grid line. Use property BWDTABKEYFILTER to associate this call with a cell of choice.	Optional	
bwdtabkeyfilter	By default the BWDTABKEYMETHOD is called if the user presses the SHIFT and TAB keys within the very first cell of the grid. Input the name of a	Optional	

	cell's VALUEPROP to associate the method call with any other column.		
Hot Keys			
hotkeys	Semicolon separated list of hot keys. A hotkey consists of a list of keys and a method name. Separate the keys by "-" and the method name again with a semicolon Example: ctrl-alt-65;onCtrlAltA;13;onEnter ...defines two hot keys. Method onCtrlAltA is invoked if the user presses Ctrl-Alt-A. Method "onEnter" is called if the user presses the ENTER key. Use the popup help within the Layout Painter to input hot keys.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

ROWINSERT Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

ROWCOPY Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			

visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

ROWDELETE Properties

Basic			
image	URL that points to the image that is shown as icon. The URL either is an absolute URL or a relative URL. If using a relative URL then be aware of that the generated page is located directly inside your project's directory. Example: "images/icon.gif" points to an icon in an images-folder that is parallel to the page itself. "../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
visibleprop	Name of the adapter parameter that provides the information if this control is displayed or not. As consequence you can control the visibility of the control dynamically.	Optional	
Online Help			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	

81 GRIDCOLHEADER - Flexible Column Headers

- Flexible Column Sizing 652
- Flexible Column Sorting 654
- GRIDCOLHEADER Properties 655
- Smart Selection of Rows - SELECTOR Control 659
- SELECTOR Properties 660

In the **example** introducing the ROWTABLEAREA2 control, the header of the grid was built by arranging certain LABEL controls, where the LABEL controls where rendered as headers:

```
<rowtablearea2 griddataprop="lines" rowcount="10" withborder="true" width="100%">
    <tr>
        ...
        <label name="First Name" asheadline="true">
        </label>
        ...
    </tr>
    <repeat>
...
...
...
```

It is also possible to use the GRIDCOLHEADER control in order to define the header of a grid. The advantages are:

- GRIDCOLHEADER controls are automatically rendered in “header style”.
- GRIDCOLHEADER controls allow to sort the grid content.
- GRIDCOLHEADER controls allow to resize a grid.

Flexible Column Sizing

Let us have a look on the following grid definition:

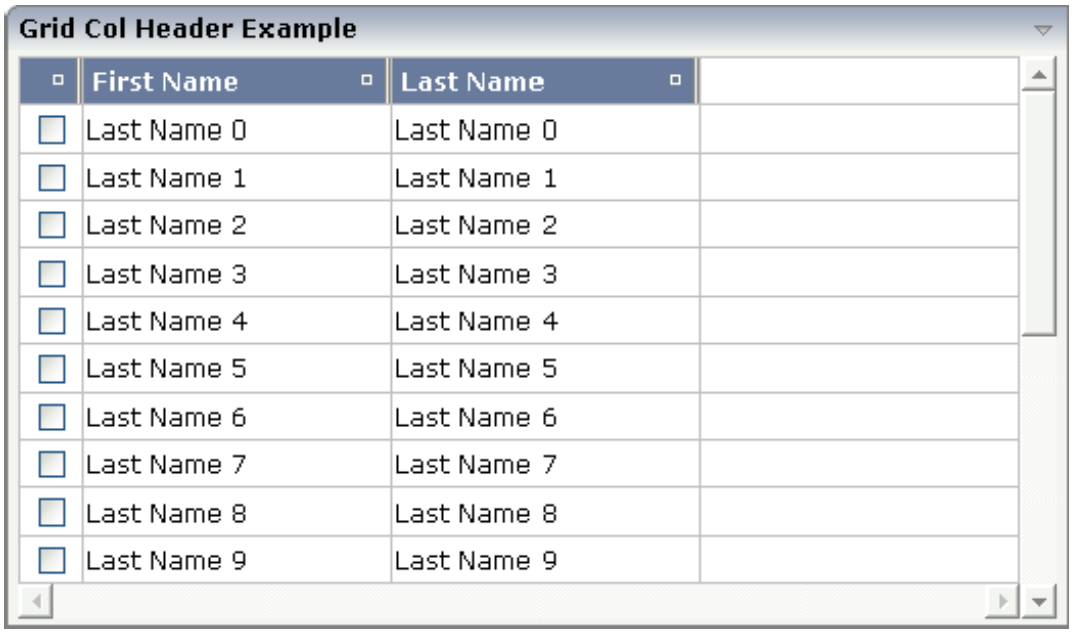
```
<rowarea name="Grid Col Header Example">
    <rowtablearea2 griddataprop="lines" rowcount="10" width="100%" withborder="true"
                   hscroll="true" firstrowcolwidths="true">
        <tr>
            <gridcolheader name=" " width="30">
            </gridcolheader>
            <gridcolheader name="First Name" width="150">
            </gridcolheader>
            <gridcolheader name="Last Name" width="150">
            </gridcolheader>
            <hdist>
            </hdist>
        </tr>
        <repeat>
            <str valueprop="selected">
                <checkbox valueprop="selected" flush="screen" width="100%" ↩
align="center">
                </checkbox>
                <field valueprop="firstName" width="100%" noborder="true"
                       transparentbackground="true">
                </field>
```

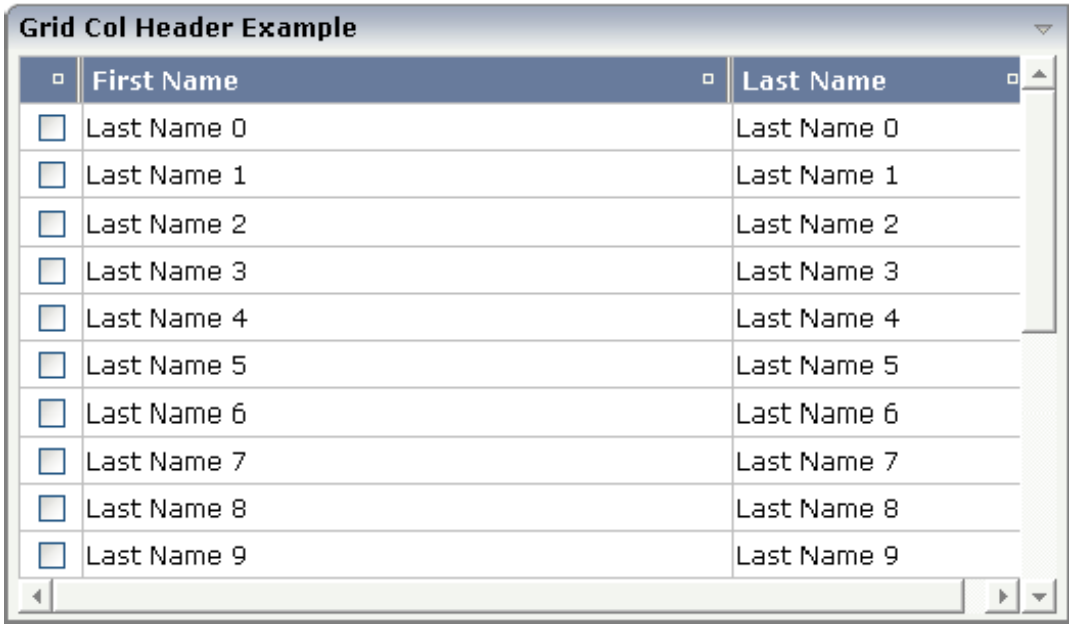
```
                <field valueprop="lastName" width="100%" noborder="true"
                       transparentbackground="true">
                </field>
                <hdist>
                </hdist>
            </str>
        </repeat>
    </rowtablearea2>
</rowarea>
```

You see:

- The ROWTABLEAREA2 definition was set to always follow the column widths of the first row. The first row of the grid is the row containing the GRIDCOLHEADER controls, this means that this row defines the column sizing for the whole grid.
- The header row of the grid is built out of GRIDCOLHEADER controls, each one specifying a name and a width.
- The header row is closed with an horizontal distance. This is quite important: if your column widths do not horizontally fill the grid, then the remaining space is typically equally distributed among the columns. Even if GRIDCOLHEADER specifies a certain width, this may still be overridden by the browser. A horizontal distance control (HDIST) at the end makes the browser assign the remaining space to the distance control, not to the GRIDCOLHEADER controls.

When the user moves the mouse over the border of the header columns, then the cursor will change and the user can change the width of the columns:





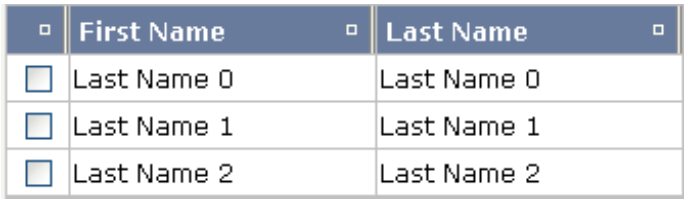
Flexible Column Sorting

The GRIDCOLHEADER allows to bind to a property which is used for sorting. The XML definition of the previous example was extended to demonstrate this:

```
<rowarea name="Grid Col Header Example">
    <rowtablearea2 griddataprop="lines" rowcount="10" width="100%" withborder="true"
                   hscroll="true" firstrowcolwidths="true">
        <tr>
            <gridcolheader name=" " width="30" propref="selected">
            </gridcolheader>
            <gridcolheader name="First Name" width="150" propref="firstName">
            </gridcolheader>
            <gridcolheader name="Last Name" width="150" propref="lastName">
            </gridcolheader>
            <hdist>
            </hdist>
        </tr>
        <repeat>
            <str valueprop="selected">
                <checkbox valueprop="selected" flush="screen" width="100%" ↩
align="center">
                </checkbox>
                <field valueprop="firstName" width="100%" noborder="true"
                       transparentbackground="true">
                </field>
                <field valueprop="lastName" width="100%" noborder="true"
                       transparentbackground="true">
```

```
                </field>
                <hdist>
                </hdist>
            </str>
        </repeat>
    </rowtablearea2>
</rowarea>
```

Each GRIDCOLHEADER control now points to the property that is referenced in the subsequent FIELD/CHECKBOX definition. The control now displays small sort icons. The user can sort the information by choosing the icon.



GRIDCOLHEADER Properties

Basic			
name	Text that is displayed inside the control. Please do not specify the name when using the multi language management - but specify a "textid" instead.	Sometimes obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Sometimes obligatory	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control	Obligatory	100 120 140 160 180 200 50% 100%

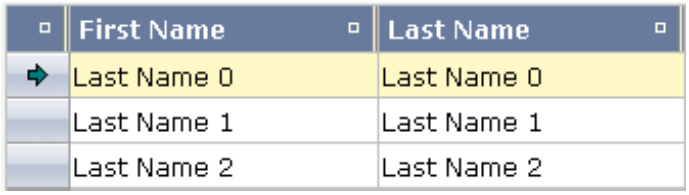
	to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		
propref	If the grid column visualizes data input the name of the property here. This property is located within the row item class. Example: if you use a FIELD or CHECKBOX control input the value of property VALUEPROP here. If the grid column does not visualize any data (e.g. you use a BUTTON control) input an unique column identifier. The PROPREF property is used as key when flushing 'column change events' to the application.	Optional	
Appearance			
title	Text that is shown as tooltip for the control. Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.	Optional	
titletextid	Text ID that is passed to the multi language management - representing the tooltip text that is used for the control.	Optional	
withsorticon	Flag that indicates if a small sort indicator is shown within the right corner of the control. Default is TRUE.	Optional	true false
withfrozencolumnsicon	Add an icon to the column header to freeze columns horizontally to the left on click.	Optional	true false
frozentrueicon	Image URL that is shown if the corresponding property value is "true".	Optional	
frozenfalseicon	Image URL that is shown if the corresponding property value is "true".	Optional	
togglestylevariant	Set a value for this attribute if you would like to apply your own sort images. Own sort images must be provided in the subfolder images of your user interface component. For a togglestylevariant ABC the name of the image files must be sort0ABC.gif, sort1ABC.gif, sort2ABC.gif.	Optional	
resizable	Set this to FALSE if the header should not be resizable. Default is TRUE.	Optional	true false
image	URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid. Use the following options to specify the URL:	Optional	

	(A) Define the URL relative to your page. Your page is generated directly into your project's folder. specifying "images/xyz.gif" will point into a directory parallel to your page. Specifying "../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project. (B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".		
nowrap	The textual content of the header is not wrapped automatically. No line break will be performed automatically by the browser. If you want the text of the header to be wrapped, set the value to "false".	Optional	true false
stylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2 VAR3 VAR4
sorttitle	Text that is shown as tooltip for the sort indicator. Either input text by using this SORTTITLE property - or use the SORTTITLETEXTID in order to define a language dependent literal.	Optional	
sorttitletextid	Text ID that is passed to the multi language management - representing the tooltip text for the sort indicator.	Optional	
movablecol	Set this attribute to TRUE if you want the columns to be movable at runtime. Default is FALSE. Please notice that only specific controls like FIELD in a grid support movable columns.	Optional	true false
textalign	Alignment of text inside the control.	Optional	left center right
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5

			10 32767
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
Binding			
visibleprop	Name of the adapter parameter that provides the information if the column is displayed or not.	Optional	
nameprop	Name of adapter parameter which dynamically provides the text that is shown inside the control.	Optional	
Comment			
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Smart Selection of Rows - SELECTOR Control

By using the SELECTOR control in combination with the STR control, you can build nice looking grids in which the user can select rows. Have a look at the following screen:



The SELECTOR control is typically is used in the leftmost column. The user can select the control with the mouse or keyboard. In case of using the control for multiple selections, the user can select multiple rows using a combination of CTRL and click or SHIFT and click.

The SELECTOR control references a boolean property inside a row object that is representing the selection state. The XML layout definition looks as follows:

```
<rowtablearea2 griddataprop="lines" rowcount="10" width="100%" withborder="true"
               hscroll="true" firstrowcolwidths="true">
    <tr>
        <gridcolheader name=" " width="30" propref="selected">
        </gridcolheader>
        <gridcolheader name="First Name" width="150" propref="firstName">
        </gridcolheader>
        <gridcolheader name="Last Name" width="150" propref="lastName">
        </gridcolheader>
        <hdist>
        </hdist>
    </tr>
    <repeat>
        <str valueprop="selected">
            <selector valueprop="selected" width="30" withlinenum="false"
                      singleselect="false">
            </selector>
            <field valueprop="firstName" width="100%" noborder="true"
                   transparentbackground="true">
            </field>
            <field valueprop="lastName" width="100%" noborder="true"
                   transparentbackground="true">
            </field>
            <hdist>
            </hdist>
        </str>
    </repeat>
</rowtablearea2>
```

You see the following:

- STR and SELECTOR are referencing the same property `selected` so that selections done by the SELECTOR control are automatically reflected in the selections of the row.
- SELECTOR is switched to allow multiple selections.
- By using the property `withlinenum`, you specify that inside the selector no line number is output. Instead, the SELECTOR is left empty if not selected, or it displays an icon if selected.

The selector simplifies programming of the grid selection a lot. When clicking the selector control, it automatically manages the referenced selection property of all rows that are managed inside the corresponding grid collection.

SELECTOR Properties

Basic			
valueprop	Name of adapter parameter that indicates the selection status of the row the selector refers to. The value is set and get by the SELECTOR control.	Optional	
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 120 140 160 180 200 50% 100%
singleselect	Indicates if the multiple lines can be selected ("false") or only one line can be selected ("true"). Default is "true".	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Binding			
valueprop	(already explained above)		
Appearance			

withlinenum	There are two usage variants: either the line number of the corresponding row is shown as content of the SELECTOR control ("true") - or nothing is shown inside ("false"). In case of selecting "true" then the line number is automatically retrieved, i.e. you do not have to specify a property on adapter side to indicate the value of the line number.	Optional	true false
image	If specifying WITHLINENUM to be "false" then a small arrow icon is shown inside the control if selecting a corresponding row. Input the URL of the icon to be shown if you do not want to use the default icon. If specifying WITHLINENUM to be "true" then the line number of selected lines is output in bold font.	Optional	
imageprop	Name of the adapter parameter which provides an image URL dynamically at runtime. The image is shown for displaying selected rows.	Optional	
alwaysshowicon	Flag that indicates if the selector shows its image - independent from whether the corresponding line is selected or not. With ALWAYSHOWICON you can show icons on unselected lines, too. For that specify WITHLINENUM to be "false" and use IMAGEPROP. Default is "false".	Optional	true false
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	

njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

82 Styling Grids

- General Hints 664
- Styling ROWTABLEAREA2, ROWTABLEAREA3 and Headers 664

In many applications you would want your grids to have the same style in the whole application. This can be achieved via adapting the corresponding style variables and/or style classes of your style sheet (*Writing the GUI Layout*).

The following gives some hints about the most frequent customizations. The NJXDEMOS contain running samples.

General Hints

When to adapt style variables and when to adapt style classes

Many controls have a headline (TEXTGRID*, ROWTABLEAREA2). If you want to change the background color for the headlines in all these controls, simply adapt the variable `@@HEADLINEBACKGROUND@@`.

If you only want to adapt the background color in one specific control, adapt the corresponding style class.

Styling ROWTABLEAREA2, ROWTABLEAREA3 and Headers

How to style the grid headers

As grid headers you usually have LABEL controls with property `asheadline=true`:

```
<tr>
  <label asheadline="true" name="Column 1" ...></label>
  <label asheadline="true" name="Column 2" ...></label>
</tr>
```

Alternatively, you are using GRIDCOLHEADER controls:

```
<tr>
  <gridcolheader name="Column1" ...></gridcolheader>
  <gridcolheader name="Column1" ...></gridcolheader>
</tr>
```

In both cases, the following applies:

- To change the headline background color for multiple controls, simply adapt the variable `@@HEADLINEBACKGROUND@@` in the **General** tab of the Style Sheet Editor tool.
- To change the headline background color only for the headers in ROWTABLEAREA2 and ROWTABLEAREA3 or if you want to do more advanced header styling, adapt the style setting:

```
LABEL
     LABELCellHeadline
```

How to style the sort icons

You can apply your own sort icons by setting the attribute `togglestylevariant` in your GRIDCOLHEADERs:

```
<gridcolheader name="Column1" togglestylevariant="ABC" ...></gridcolheader>
```

As value for `togglestylevariant` use a simple short string. Mostly you would use a subset from your style sheet name.

Copy your own sort icon images to a subfolder named *images* of your user interface component. Adhere to the following naming conventions for the file names:

Sort status	File name	Example
unsorted	sort0<togglestylevariant>.gif	sort0ABC.gif
sort descending	sort1<togglestylevariant>.gif	sort1ABC.gif
sort ascending	sort2<togglestylevariant>.gif	sort2ABC.gif

How to style (background) colors of your grid cells

A simple way to adapt colors for the grid cells is: In the TR or STR of your REPEAT control set the `withalterbackground` property to true:

```
<repeat>
        <str withalterbackground="true" ...>
                <field valueprop="somefield" ...>
```

Now you can simply adapt the variables `@@EVENCELLBACKGROUND@@` and `@@ODDCELLBACKGROUND@@` using the Style Sheet Editor tool. This will also change the background color in TEXTGRID* controls to the same value.

To change the background color only for ROWTABLEAREA2 and ROWTABLEAREA3 or if you want to do more advanced row styling, adapt the style settings:

```
STR
   STRRowEven
   STRRowOdd
```

How to style (background) colors of your header

If your header row consists only of GRIDCOLHEADER controls, you would usually style your header cells as described above. Sometimes you have additional controls like a TOGGLE attribute

in the header row of your grid. In this case, you want to set a specific background color for the whole header row including the cells with the TOGGLE controls.

To style the header row in general, set the property `asheadline` in the TR container of the header as follows:

```
<tr asheadline="true">... </tr>
```

Then you can adapt the style in the style class:

```
TR
   TRHeadline
```

How to set common style attributes like padding for all cells in a grid

If you want to set the same padding value for all cells in the grid adapt the style setting:

```
TABLEAREA
   TABLEAREABorder>tbody>tr>td
```

How to set different style attributes like padding for all header and content cells

If you have resizable headers you usually do not want to have the same style settings of your content cells in the header. For instance, you do not want to set padding attributes in the header but would like to have general padding settings in the content cells.

To style the content classes use the style classes under STR in the Stylesheet Editor tool. For example you could set padding-left and padding-right style attributes for the content cells in the following style classes:

```
STR
   STRRowSelected
   STRRowUnSelected
```

For the header cells you would set different values for the padding-left and padding-right attributes in the following style class:

```
TR
   TRHeadline
```

Since padding values are not inherited down an HTML table in HTML5, you need to additionally set the values for the `padding-left` and `padding-right` attributes to `inherit` in the following style class:

```
TABLEAREA
   TABLEAREABorder>tbody>tr>td
```

That way, you are actually setting different values for the different rows, thus making sure these different values are inherited down the cells underneath the rows.

How to customize the border style of the grid

To adapt general table settings such as the border for the grid, adapt the style classes under TABLEAREA.

V Working with Trees

This part shows you how to work with trees and tree nodes. The information is organized under the following headings:

Basics

TREENODE3 in Control Grid (ROWTABLEAREA2)

CLIENTTREE

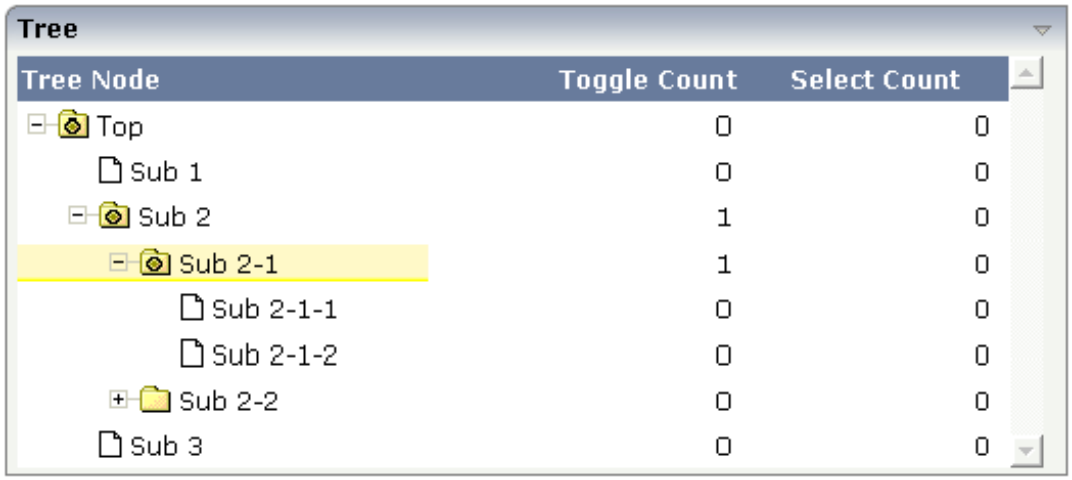
83 Basics

- Types of Trees 672
- When to Use Which Type 673

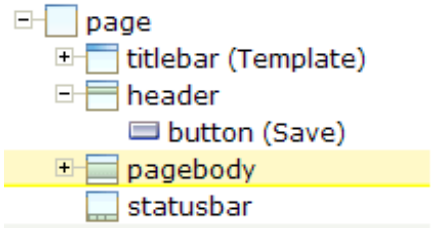
Types of Trees

The following controls are available for building trees:

- **TREENODE3**
  This control displays a single tree node. It can be put into the normal control grid (ROWTABLEAREA2), and can consequently be combined with any other control (for example, FIELD, TEXTOUT, etc.).



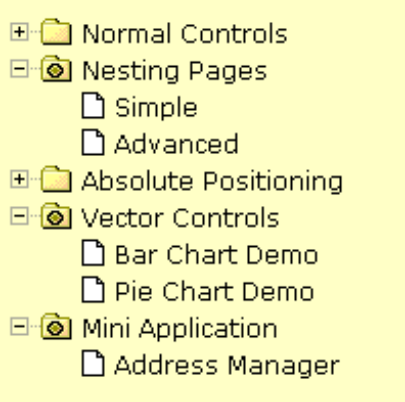
  Of course, you do not have to combine it with other controls. You can also use it “stand-alone” inside a ROWTABLEAREA2 grid:



  As with the normal ROWTABLEAREA2 management, only these items are transferred from the server to the client which are currently visible. Items which are collapsed or which are not in the visible area of the client, are not transferred.

  All scrolling of items and all toggling of items (opening/collapsing) goes through the server.

- **CLIENTTREE**
  This control represents a whole tree. You cannot add further controls into the tree node lines.



The data which is displayed inside the tree is transferred from the server to the client in one step - always the whole tree. The data is transferred when opening a page or when the tree data in the server is updated.

All scrolling of items and all toggling of items (opening/collapsing) is done in the client without going back to the server.

When to Use Which Type

Use the TREENODE3 control inside the control grid ROWTABLEAREA2 in the following cases:

- High number of tree nodes.
- Tree nodes are not loaded from the beginning, but step by step.
- Data in the tree is exchanged/updated quite often.

Use the CLIENTTREE control in the following cases:

- Low number of tree nodes (100).
- High interactivity requirements for toggling nodes.
- Data in the tree is rather static. It is loaded once into the client, and afterwards it is not changed anymore.

Example: in the Application Designer environment, the tree controls are used in the following way:

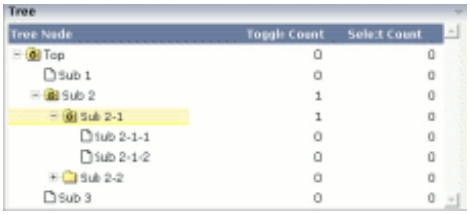
- In the workplace, a CLIENTTREE is loaded: the number of nodes is quite low, the tree represents a menu which is rather static.
- In the Layout Painter, a TREENODE2 in a ROWTABLEAREA2 is used for representing the XML control tree: the number of items can be quite high, the update rate of the tree data is very high.

84 TREENODE3 in Control Grid (ROWTABLEAREA2)

- Example 676
- Adapter Interface 677
- Built-in Events 677
- Properties 677

Example

The following image shows an example for a tree management:



The grid contains three columns: the first column shows the tree node, the other two columns display some text information.

The XML layout definition is:

```
<rowarea name="Tree">
    <rowtablearea2 griddataprop="treeGridInfo" rowcount="8" width="500" ↩
withborder="false">
        <tr>
            <label name="Tree Node" width="200" asheadline="true">
            </label>
            <label name="Toggle Count" width="100" asheadline="true"
                   labelstyle="text-align:right">
            </label>
            <label name="Select Count" width="100" asheadline="true"
                   labelstyle="text-align:right">
            </label>
        </tr>
        <repeat>
            <tr>
                <treenode3 width="200" withplusminus="true"
                           imageopened="images/fileopened.gif"
                           imageclosed="images/fileclosed.gif"
                           imageendnode="images/fileendnode.gif">
                </treenode3>
                <textout valueprop="toggleCount" width="100" align="right">
                </textout>
                <textout valueprop="selectCount" width="100" align="right">
                </textout>
            </tr>
        </repeat>
    </rowtablearea2>
</rowarea>
```

You see that the TREENODE3 control is placed inside the control grid just as a normal control. There are certain properties available which influence the rendering: in the example, the name of

the tree node images is statically overwritten. The flag `withplusminus` is set to true - consequently, small "+"/"-" icons are placed in front of the node.

Adapter Interface

In the parameter data area of the adapter, the tree data is represented by the following data structure:

```
1 TREEGRIDINFO (1:*)
2 DRAGINFO (U) DYNAMIC
2 DROPINFO (U) DYNAMIC
2 LEVEL (I4)
2 OPENED (I4)
2 SELECTCOUNT (U) DYNAMIC
2 TEXT (U) DYNAMIC
2 TOGGLECOUNT (U) DYNAMIC
```

Built-in Events

value-of-griddataprop.reactOnSelect
value-of-griddataprop.reactOnToggle

Properties

Basic			
width	Width of the control. There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify	Optional	1 2 3 int-value

	a width then the rendering result may not represent what you expect.		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withplusminus	If set to "true" then +/- Icons will be rendered in front of the tree items.	Optional	true false
withlines	If set to "true" then the tree elements are connected with one another by gray lines. Please pay attention: PAGE layouts (Java), if switching this property to "true" then you have to create the instance of your server side TREECollection object with a special constructor: Example: TREECollection m_tree = new TREECollection(true)	Optional	true false
withtooltip	If set to "true" then the text of an item is also available as tool tip. Use this option in case you expect that the horizontal space of the item will not be sufficient to display the whole text of the item.	Optional	true false
withtextinput	If set to "true" then the tree node can also be edited. Editing is started when the user double clicks the node. The text that is input is passed into the property "text" which is implemented in the default NODEInfo implementation.	Optional	true false
imageopened	Image of a tree node that has subnodes and that is currently showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageclosed	Image of a tree node that has subnodes and that is currently not showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageendnode	Image of a tree node that is an end node (leaf node). The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
singleselect	If set to "true" then only one item can be selected. If set to "false" then multiple icons can be selected.	Optional	true false
directselectevent	Event that represents a tree node selection. A tree node selection is done when the user clicks/doubleclicks on the tree node text.	Optional	ondblclick onclick

	In this case the select() method is called in the corresponding node object on server side.		
directselectelement	If set to "textonly" only user clicks on the tree node text will select the node. If set to "allspace" also user clicks outside the area occupied by the node text will select the node.	Optional	textonly allspace
selectionstylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
textstylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
pixelshift	Number of pixels that each hierarchy level is indented. If not defined then a standard is used.	Optional	1 2 3 int-value
pixelshiftendnode	Number of pixels that end nodes are indented. If not defined then a standard is used.	Optional	1 2 3 int-value
colspan	Column spanning of control. If you use TR table rows then you may sometimes want to control the number of columns your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns.	Optional	1 2 3

	The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		4 5 50 int-value
rowspan	Row spanning of control. If you use TR table rows then you may sometimes want to control the number of rows your control occupies. By default it is "1" - but you may want to define the control to span over more than one columns. The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.	Optional	1 2 3 4 5 50 int-value
pixelheight	Height of the control in pixels.	Optional	1 2 3 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
Binding			
imageprop	Name of an adapter parameter that provides for a image for the tree node. Each node may provide for its own image, e.g. dependent on the type of node. If the adapter property passes back an empty string, then the image is taken from the static definitions, which you can create	Optional	

	in parallel using the properties IMAGEOPENED, IMAGECLOSED and IMAGEENDNODE.		
focusedprop	Name of the adapter parameter that indicates if the row receives the keyboard focus. If more than one lines are returning "true", the first of them is receiving the focus.	Optional	
flush	Flush behaviour when using the possibility of having editable tree nodes. If double clicking on the tree node then you can edit its content. The FLUSH property defines how the browser behaves when leaving the tree node's input field: If not defined ("") then nothing happens - the changed tree node text is communicated to the server side adapter object with the next roundtrip. If defined as "server" then immediately when leaving the field a roundtrip to the server is initiated - in case you want your adapter logic to directly react on the item change. If defined as "screen" then the changed tree node text is populated inside the page inside the front end.	Optional	screen server
flushmethod	When the data synchronization of the control is set to FLUSH="server" then you can specify an explicit event to be sent when the user updates the content of the control. By doing so you can distinguish on the server side from which control the flush of data was triggered.	Optional	
tooltipprop	Name of the adapter parameter that provides for a text that is shown if the user moves the mouse over the tree item (tooltip).	Optional	
validdraginfosprop	Name of an adapter parameter that contains a comma separated list of valid drag informations.	Optional	
Drag and Drop			
enabledrag	If set to true then drag and drop is enabled within the tree.	Optional	true false

85 CLIENTTREE

- Example 684
- Adapter Interface 684
- Built-in Events 685
- Properties 685

Example

The following example shows a simple client tree:



The XML layout definition is:

```
<rowarea name="Clienttree">
    <clienttree treecollectionprop="tree" height="200" withplusminus="true"
                treestyle="background-color:#FEFEEE">
    </clienttree>
</rowarea>
```

In this example, the client tree is directly put as row into the ROWAREA container. The property `treecollectionprop` contains a reference to the property `tree` which contains the net data of the tree. With the property `treestyle`, an explicit background color is set.

Adapter Interface

In the parameter data area of the adapter, the tree data is represented by the following data structure:

```
1 TREE (1:*)
2 LEVEL (I4)
2 OPENED (I4)
2 SELECTED (L)
2 TEXT (U) DYNAMIC
```

Built-in Events

value-of-treecollectionprop.reactOnContextMenuRequest
value-of-treecollectionprop.reactOnSelect
value-of-treecollectionprop.reactOnToggle

Properties

Basic			
treecollectionprop	Name of the adapter parameter that represents the control in the adapter.	Optional	
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	100 150 200 250 300 250 400 50% 100%
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withplusminus	If set to "true" then +/- Icons will be rendered in front of the tree items.	Optional	true false
withtooltip	If set to "true" then the text of an item is also available as tool tip. Use this option in case you expect that the horizontal space of the item will not be sufficient to display the whole text of the item.	Optional	true false

selectionvisible	If set to "true" then the clicked item will also marked with a certain background color. The background color is defined by the style sheet settings.	Optional	true false
singleselect	If set to "true" then only one item can be selected. If set to "false" then multiple icons can be selected.	Optional	true false
imageopened	Image of a tree node that has subnodes and that is currently showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageclosed	Image of a tree node that has subnodes and that is currently not showing its nodes. The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
imageendnode	Image of a tree node that is an end node (leaf node). The image either is defined statically by this property or also may be defined dynamically - see the corresponding properties defined with this control.	Optional	
treestyle	Style (following cascading style sheet definitions) that is directly passed to the background area of the client tree. You can manipulate e.g. the colour of the tree's background. The style can also be set dynamically by specifying the property TREESTYLEPROP.	Optional	
selectionstylevariant	Some controls offer the possibility to define style variants. By this style variant you can address different styles inside your style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen. Purpose: you can set up style variants in the style sheet definition and use them multiple times by addressing them via the "stylevariant" property. CIS currently offers two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!	Optional	VAR1 VAR2
hscroll	Definition of the horizontal scrollbar's appearance. You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".	Optional	auto scroll hidden
pixelshift	Number of pixels that each hierarchy level is indented. If not defined then a standard is used.	Optional	1 2

			3 int-value
pixelshiftendnode	Number of pixels that end nodes are indented. If not defined then a standard is used.	Optional	1 2 3 int-value
tabindex	Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.	Optional	-1 0 1 2 5 10 32767
withleftpadding	Flag that indicates if the control has a 10 pixel padding on left side. Default is true.	Optional	true false
Binding			
treecollectionprop	(already explained above)		
dynamicloading	If set to "true" then you indicate to the tree control that not all tree information may be loaded when initializing the tree (i.e. the tree collection on server side). As consequence the tree control will pass the "toggle-event" to the server - in case the subnodes of a certain nodes are not yet loaded. In the case the toggle event is passed to the server, the method onToggle() is called inside the tree item.	Optional	true false
imageopenedprop	Name of the adapter parameter that provides the image URL which is shown for opened tree nodes or end tree nodes. The value may be different from tree node to tree node. Each tree node may have an own image.	Optional	
imageclosedprop	Name of the adapter parameter that provides for the image URL which is shown for closed tree nodes. The value may be different from tree node to tree node. Each tree node may have an own image.	Optional	
treestyleprop	Name of the adapter parameter that dynamically provides for a style value that is passed to the control's area	Optional	

	(background of the client tree). You can as consequence e.g. define the background-color of the tree dependent on your server side logic.		
treeclassprop	Name of the adapter parameter that passes back the name of a style sheet class that is taken to render the client tree's background area. - Similar to the property TREESTYLEPROP, but now a style class is passed, not the style itself.	Optional	
tooltipprop	Name of the adapter parameter that provides for a text that is shown if the user moves the mouse over the tree item (tooltip).	Optional	
oncontextmenumethod	Name of the event that is sent to the adapter when the user presses the right mouse button in an empty area of the client tree.	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user double clicks.	Optional	
directselectevent	Event that represents a tree node selection. A tree node selection is done when the user clicks/doubleclicks on the tree node text. In this case the select() method is called in the corresponding node object on server side.	Optional	ondblclick onclick
focusedprop	Name of the adapter parameter that indicates if the row receives the keyboard focus. If more than one lines are returning "true", the first of them is receiving the focus.	Optional	
Drag and Drop			
enabledrag	If set to true then drag and drop is enabled within the tree.	Optional	true false

VI Working with Menus

Menus are used to arrange a number of functions in a structured way.

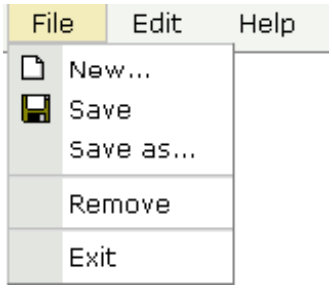
The information provided in this part is organized under the following headings:

Types of Menus
MENU
DLMENU
XCIPOPUPMENU - Enable Context Menus
Styling Menus

86 Types of Menus

The following menu controls are available:

- **MENU**
  This is the typical drop-down menu:

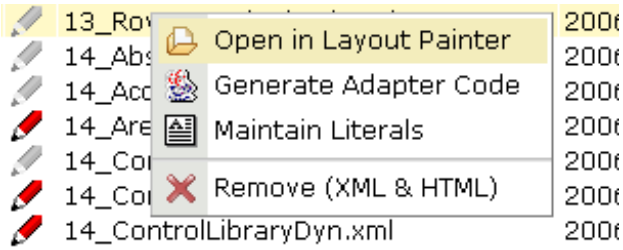


- **DLMENU**
  This is a double-line menu representing a two-level hierarchy. It can be found quite often in web applications.



  When clicking an item in the first line, the corresponding subitems are shown in the second line.

- **Context Menu**
  This is a menu which appears in certain controls (tree controls, grid controls) when the user presses the right mouse button.



All menu controls are dynamically configured by the application. This means:

- The structure of the menu and its menu nodes is not statically defined but is dynamically controlled by the application through adapter parameters. For example, you can build a personalized menu taking the user's rights into consideration.
- Menu information can be dynamically updated during runtime.

87 MENU

- Example 694
- Adapter Interface 695
- Built-in Events 695
- Properties 695

Example

The example looks as follows:



When clicking on a menu item for which a function has been defined, then the name of the function is displayed in the status bar.

The XML layout definition is:

```
<page model="Menue_01_Adapter">
    <titlebar name="Menu Demo">
    </titlebar>
    <header align="left" withdistance="false">
        <menu menucollectionprop="menuData" width="100">
        </menu>
    </header>
    <pagebody>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
</page>
```

In this example, the menu is embedded in the header. By the property `menucollectionprop`, it is bound to the adapter property `menuData`.

Adapter Interface

```
1 MENUDATA (1:*)
2 ID (U) DYNAMIC
2 IMAGEURL (U) DYNAMIC
2 LEVEL (I4)
2 METHOD (U) DYNAMIC
2 OPENED (I4)
2 TEXT (U) DYNAMIC
1 SELMENUITEM (U) DYNAMIC
```

Built-in Events

items.reactOnSelect

Properties

Basic			
menucollectionprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
width	Width of the control.	Optional	100

	There are three possibilities to define the width: (A) You do not define a width at all. In this case the width of the control will either be a default width or - in case of container controls - it will follow the width that is occupied by its content. (B) Pixel sizing: just input a number value (e.g. "100"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a width this control can reference. If you specify this control to have a width of 50% then the parent element (e.g. an ITR-row) may itself define a width of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.		120 140 160 180 200 50% 100%
height	Height of the control. There are three possibilities to define the height: (A) You do not define a height at all. As consequence the control will be rendered with its default height. If the control is a container control (containing) other controls then the height of the control will follow the height of its content. (B) Pixel sizing: just input a number value (e.g. "20"). (C) Percentage sizing: input a percentage value (e.g. "50%"). Pay attention: percentage sizing will only bring up correct results if the parent element of the control properly defines a height this control can reference. If you specify this control to have a height of 50% then the parent element (e.g. an ITR-row) may itself define a height of "100%". If the parent element does not specify a width then the rendering result may not represent what you expect.	Optional	
toggleimage	URL of the image that is shown on the right end of a menu item, if this item contains subitems. If not explicitly defined then a default icon is used.	Optional	
toggleimageprop	Name of the adapter parameter that provides a URL that defines the toggle image. The toggle icon is shown on the right end of a menu item that has subitems.	Optional	
menustyle	CSS style definition that is directly passed into this control. With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are: border: 1px solid #FF0000 background-color: #808080 You can combine expressions by appending and separating them with a semicolon.	Optional	

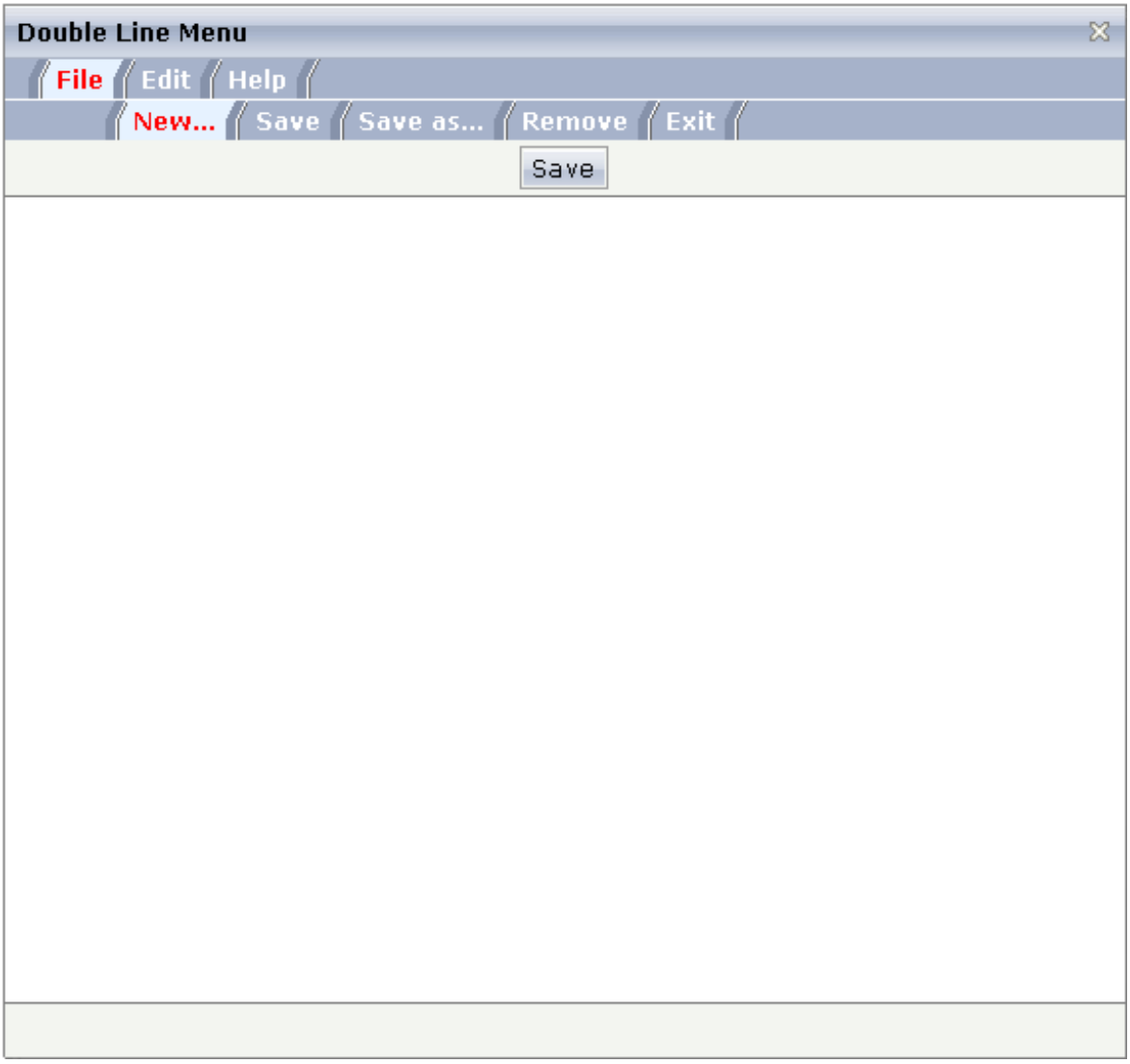
	Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
menustyleprop	Name of the adapter parameter that dynamically provides explicit style information for the control.	Optional	
withinactivenodes	Set the value of this property to true if you want to disable menu items. In this case, the Natural adapter is generated with an additional INACTIVE field.	Optional	true false

88 DLMENU

- Example 700
- Adapter Interface 701
- Built-in Events 701
- Properties 701

Example

The example looks as follows:



A double-line menu is displayed. When selecting a menu item, then its text is written to the status bar.

The XML layout definition is:

```
<page model="menue_02_dl_Adapter">
    <titlebar name="Double Line Menu">
    </titlebar>
    <dlmenu menuprop="menuData">
    </dlmenu>
    <header withdistance="false">
        <button name="Save">
        </button>
    </header>
    <pagebody>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
</page>
```

The DLMENU control is positioned directly following the title bar. In its property `menuprop`, it holds a binding to the property `menuData`.

Adapter Interface

```
1 ITEMS (1:*)
2 LEVEL (I4)
2 METHOD (U) DYNAMIC
2 TEXT (U) DYNAMIC
```

Built-in Events

items.onSelectSubItem

Properties

Basic			
menuprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
align	Horizontal alignment of the control's content. Default is "center".	Optional	left center right

onlyoneline	If set to "true" then the DLMENU control only contains its top line - there is no second line below. Default is "false".	Optional	true false
cellseparatoronly	If set to "true" then only a very thin cell separator is added between two menu items. Otherwise the separation is rendered explicitly.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Miscellaneous			
testtoolid	Use this attribute to assign a fixed control identifier that can be later on used within your test tool in order to do the object identification	Optional	

89 XCIPOPUPMENU - Enable Context Menus

- Example 705
- Adapter Interface 706
- Built-in Events 707
- Properties 707

With an XCIPOPUPMENU control, you enable the usage of context menus on a page. The application creates the contents of the context menus dynamically at execution time, in response to certain events. There is only one instance of XCIPOPUPMENU needed in each page.

Context menus are supported on the page level and by the following controls:

- **NATPAGE**
- **TEXTGRID2**
- **TEXTGRIDSSS2**
- **MGDGRID**
- **ROWTABLEAREA3**
- **STR**
- **TREENODE3**
- **CLIENTTREE**
- **FIELD**/**FIELD3**
- **METHODLINK**

The following events are raised when the user right-clicks in corresponding areas of the page:

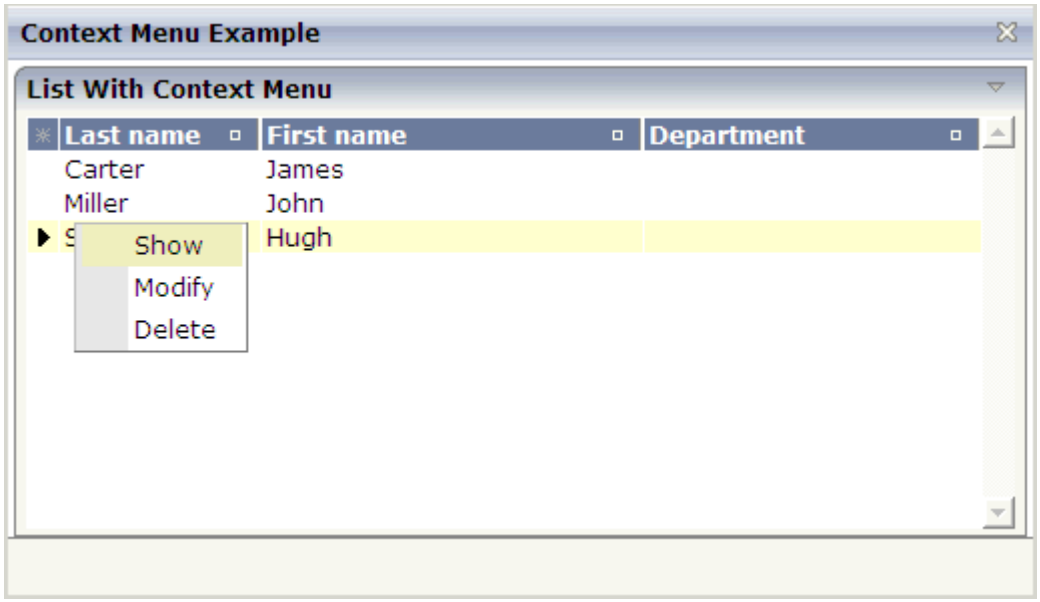
- When the user right-clicks in a non-empty line in a grid or tree, the event *value-of-griddataprop*`.reactOnContextMenuRequest` or *value-of-treecollectionprop*`.reactOnContextMenuRequest` is raised.
- When the user right-clicks in an empty line in a grid or tree, the event defined in the property `contextmenumethod` of the grid or tree is raised.
- When the user right-clicks in a FIELD control for which "myfield" has been defined with the `valueprop` property, the event `reactOnContextMenuMyfield` or the event defined in the property `contextmenumethod` of the FIELD control is raised (see the description of the FIELD control for more information on the properties `contextmenu` and `contextmenumethod`).
- When the user right-clicks elsewhere in the page, the event defined in the `contextmenumethod` of the page is raised.

In the event handler of these events, you do no have to necessarily open a context menu; you can also start other operations, if this makes sense. But in order to open a context menu, you need to fill the structure generated for the XCIPOPUPMENU control, which is described below.

If the user selects one of the context menu items, the event `xcipopupmenu.reactOnSelect` is raised.

Example

The following screen displays a grid control with several rows. It uses the XCIPOPUPMENU control to show a context menu when the user right-clicks on a row. It shows a different context menu when the user right-clicks in an empty area of the grid and yet another one when the user right-clicks elsewhere in the page.



The XML layout definition contains the following:

```
<natpage>
    <xcipopupmenu>
    </xcipopupmenu>
...
</natpage>
```

The example Natural code is contained in the Natural for Ajax demos as program `CTRCTX-P`.

Adapter Interface

```
1 XCIPOPUPMENU
2 MENUNODE (1:*)
3 ID (A) DYNAMIC
3 IMAGE (A) DYNAMIC
3 LEVEL (I4)
3 REFERENCE (A) DYNAMIC
3 TEXT (A) DYNAMIC
2 SELECTEDREFERENCE (A) DYNAMIC
```

A menu is reflected by a tree of menu nodes. Each menu node is represented by an `ID`, a `TEXT`, an optional `IMAGE` and a `REFERENCE` value. When the user selects a menu item, the `REFERENCE` value of that menu item is then returned in the parameter `SELECTEDREFERENCE`.

To include a separator in a context menu, you define "&SEPARATOR" as the `TEXT` value. Example:

```
RESIZE ARRAY MENUNODE TO (1:3)
XCIPOPUPMENU.ID(1) := '1'
XCIPOPUPMENU.TEXT(1) := 'Open'
XCIPOPUPMENU.LEVEL(1) := 1
XCIPOPUPMENU.ID(2) := '2'
XCIPOPUPMENU.TEXT(2) := '&SEPARATOR'
XCIPOPUPMENU.LEVEL(2) := 1
XCIPOPUPMENU.ID(3) := '3'
XCIPOPUPMENU.TEXT(3) := 'Close'
XCIPOPUPMENU.LEVEL(3) := 1
```

If you want to disable menu items, you have to set the value of the `withinactivenodes` property to "true". In this case, the adapter is generated with an additional `INACTIVE` field:

```
1 XCIPOPUPMENU
2 MENUNODE (1:*)
3 ID (A) DYNAMIC
3 IMAGE (A) DYNAMIC
3 INACTIVE (L)
3 LEVEL (I4)
3 REFERENCE (A) DYNAMIC
3 TEXT (A) DYNAMIC
2 SELECTEDREFERENCE (A) DYNAMIC ↩
```

By default, the value of the `INACTIVE` field is "false". If you set it to "true", the corresponding menu item is shown as inactive, that is, it is greyed out and cannot be selected.

Built-in Events

xcipopupmenu.reactOnSelect

Properties

Basic			
withinactivenodes	Set the value of this property to true if you want to disable menu items. In this case, the Natural adapter is generated with an additional INACTIVE field.	Optional	true false

90 Styling Menus

- Shared and Unshared Style Classes in Menu Controls 710

In many applications you are using different kind of menu controls, for instance context menus and drop down menus. The different menus have common style classes. This allows for applying the same look to different controls. For the styling of the menus, adapt the style variables and/or style classes of your style sheet using the Style Sheet Editor tool (see Ajax Developer documentation). The following provides some hints concerning the style classes shared by several different menu controls as well as the unshared classes.

Shared and Unshared Style Classes in Menu Controls

Both menu controls support separating menu items by a separator. A separator consists of 4 rows:

Row	Description	Style Classes
Row 1	Space above the separator	`MENUSeparatorImgAboveCell`, `MENUSeparatorLabelAboveCell`
Row 2	Separator first line	`MENUSeparatorFirstRow`
Row 3	Separator second line	`MENUSeparatorSecondRow`
Row 4	Space below the separator	`MENUSeparatorImgBelowCell`, `MENUSeparatorLabelBelowCell`

Context and drop down menus use individual style classes and also share some common style classes. They for example share the style classes for the separator mentioned above. The following table lists the most common shared and unshared style classes:

Class	Context Menu	Drop Down Menu	Description
`MENUItemTable`	X	X	Is applied to the `<table>` that encases the menu items.
`MENUItemImageCell`	X	X	Each menu item row consists of an image and a label cell. This style class is applied to the image cell (`<td>` element).
`MENUItemImageCellRollOver`	X	X	When rolling with the mouse over a menu item, this style class replaces the `MENUItemImageCell` style class.
`MENUItemLabelCell`	X	X	Each menu item row consists of an image and a label cell. This style class is applied to the label cell (`<td>` element).
`MENUItemLabelCellRollOver`	X	X	When rolling with the mouse over a menu item, this style class replaces the `MENUItemLabelCell` style class.
`MENUSeparatorImgAboveCell`	X	X	In a separator, this style class is applied to the image (left) cell of the space above (Row 1).
`MENUSeparatorLabelAboveCell`	X	X	In a separator, this style class is applied to the label (right) cell of the space above (Row 1).

Class	Context Menu	Drop Down Menu	Description
MENUSeparatorFirstRow	X	X	In a separator, this style class is applied to the image and label cell of the separator first line (Row 2).
MENUSeparatorSecondRow	X	X	In a separator, this style class is applied to the image and label cell of the separator second line (Row 3).
MENUSeparatorImgBelowCell	X	X	In a separator, this style class is applied to the image (left) cell of the space below (Row 4).
MENUSeparatorLabelBelowCell	X	X	In a separator, this style class is applied to the label (right) cell of the space below (Row 4).
MENUTextCell	X		Use this style class to define the text style of context menu items. The text of a menu item is rendered within an <a> element. This <a> element is embedded in a <td> element. This style class is applied to that <td> element.
MENUTextCellRollOver	X		When rolling the mouse over an item within a context menu, this style class replaces the MENUTextCell style class.
MENUTextCellInactive	X		When a context menu item is set to inactive (i.e. not selectable), this style class is applied to the <td> element, which renders the text.
MENUTop		X	In contrast to context menus, drop down menus have a header line, which is always visible. This header line displays the top nodes. Each top node is rendered within a <table> element. This style class is applied to that <table> element.
MENUTopRollOver		X	When rolling the mouse over a top node, this style class replaces the MENUTop style class.
MENUTopPressed		X	When clicking on a top node, this style class replaces the MENUTopRollOver style class.
MENUTopTextCell		X	Use this style class to define the text within header lines. The text of a top node is rendered within an <a> element. This <a> element is embedded in a <td> element. This style class is applied to that <td> element.
MENUTopTextCellRollOver		X	When clicking on a top node, this style class replaces the MENUTopTextCell style class.
MENUDropDownTextCell		X	Use this style class to define the text style of drop down items other than top nodes. The text of a menu item is rendered within an <a> element. This <a> element is embedded in a <td> element. This style class is applied to that <td> element.

Class	Context Menu	Drop Down Menu	Description
MENUDropDownTextCellRollOver		X	When rolling the mouse over an item within a drop down table, this style class replaces the MENUDropDownTextCell style class.
MENUDropDownTextCellInactive		X	When a menu item other than a top node is set to inactive (i.e. not selectable), this style class replaces the MENUDropDownTextCell style class.

VII Non-Visual Controls and Hot Keys

This part describes some controls that do not have any visual effect to your screen, but provide some client functions to be applied to your page.

The information provided in this part is organized under the following headings:

AUTOCOMPLETE
TIMER
XCIDATADEF - Data Definition
XCICONTEXT
NJX:XCIOPENPOPUP
NJX:XCILIVINGPOPUP
Extended Hot Key Management
Function Key Handling
NJX:OBJECTS
NJX:SESSIONPARAMS
NJX:REQUESTCONTEXT
NJX:TRIGGEREVENT

91 AUTOCOMPLETE

- General Information 716
- Example 717
- Data Sources for Populating the Values 717
- Properties 720

The AUTOCOMPLETE control adds additional functionality to a **FIELD** or BMOBILE:FFIELD control. It supports the selection of a value from a pre-populated list of values while typing. The list of values can be populated from different data sources. On selection, additional values can be displayed in other controls.

General Information

To add autocomplete functionality to a FIELD or BMOBILE:FFIELD control, you drag an AUTOCOMPLETE control to your layout page. This AUTOCOMPLETE control can be referenced as data source from within multiple FIELD or BMOBILE:FFIELD controls. Adding a reference to an AUTOCOMPLETE data source for a FIELD or BMOBILE:FFIELD control is done via the `id` property of the AUTOCOMPLETE control and the `autocompleteref` property of the FIELD or BMOBILE:FFIELD control.

```
<autocomplete id="myautocompleteid" ...></autocomplete>
<field valueprop="myinputcontrol" autocompleteref=" myautocompleteid">
</field>
```

In the AUTOCOMPLETE control, you define from which data source the value list will be populated. The `source` property specifies the kind of data source and the `sourcelocation` property a concrete source depending on the kind of data source. You can specify values for the source and source location either at design time or dynamically at runtime. For the latter, the properties `sourceprop` and `sourcelocationprop` are supported.

Populating the value list is done while typing into the FIELD or BMOBILE:FFIELD control. With the property `minlength`, you can specify the number of characters after which the value list is to be populated. With the property `delay`, you can specify a delay between a keystroke and the population of the list.

In the example below, the value list is not populated unless at least three characters have been inserted. The population of the list with the values from the data source starts 20 milliseconds after the third character has been inserted.

```
<autocomplete id="myautocompleteid" minlength="3" delay="20" ...></autocomplete>
```

The size of the drop-down box which renders the value list can be customized with the property `rows`. For example, if `rows="5"` is specified, the corresponding drop-down box will have a height of 5 rows. If the value list contains more values, a scrollbar is shown.

Example

Examples which show the usage of the AUTOCOMPLETE control are provided in the Natural for Ajax demos: programs `FDAUTO-P` and `FFAUTO-P`.

Data Sources for Populating the Values

The kind of data source you are using highly depends on the number of total values, whether the values are static or change dynamically on your application context and whether the application is showing internal application data or data from an external service. Also in some applications it is sufficient to populate a simple value list, other applications need to show additional data for each value. The range of supported data sources is from simple static string values to services which deal with thousands of values.

The following kinds of data sources are available.

- String
- File
- Url

String

Choose this data source if you simply want to show a small list of simple values. The values can either be defined at design time or at runtime via the `sourcelocation` property or the `sourcelocationprop` property. In the example below, the total number of values is four. The possible values are defined at design time.

```
<autocomplete id="mycitiesid" source="string"
sourcelocation="Bielefeld;Darmstadt;Frankfurt;Karlsruhe"></autocomplete>
```

When the user types the character "D", all values for which the first character matches "D" are offered by default. In the above example, the value list will contain the item "Darmstadt". We are calling this “match mode”. In addition to this default match mode called "start", another match mode called "all" is supported. When the match mode is "all" and the user types the character "D", the value list will be populated with all items containing a "D". In the above example, these are "Darmstadt" and "Bielefeld". The match mode can be specified via the property `matchmode`. For example:

```
<autocomplete id="mycitiesid" source="string"
sourcelocation="Bielefeld;Darmstadt;Frankfurt;Karlsruhe" ↩
matchmode="all"></autocomplete>
```

File

Choose this data source if you have a medium number of total values and either the offered values are fixed or you have a small number of variations of the total list. The "file" data source is more powerful than the "string" data source because you can show additional data for the offered values.

The data must be stored in a file with simple JSON format (see also *https://www.w3schools.com/js/js_json_intro.asp*) and must be accessible from within the web application. The following formats are supported:

Simple

Example:

```
[
"Bielefeld",
"Darmstadt",
"Frankfurt",
"Karlsruhe"
]
```

The offered functionality is the same as for String (see above). The advantage is that the values are neither in the page layout nor in the Natural code. This allows for simple replacement of the files when the offered values change without touching any application code.

Value

Example:

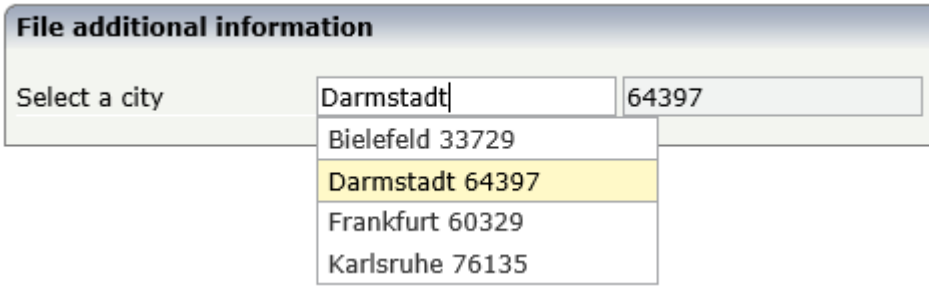
```
[
{ "code": "33729", "value": "Bielefeld" },
{ "code": "64397", "value": "Darmstadt" },
{ "code": "60329", "value": "Frankfurt" },
{ "code": "76135", "value": "Karlsruhe" }
]
```

Each offered item has a "value" for populating the value list and can have additional data like "code" as in the above example. The same match modes are supported as for String (see above). Choose this format if you want to show additional data for the offered values.

Showing additional data is done by using the properties `autocompletedisplayname` and `autocompletedisplayref` in the FIELD or BMOBILE:FFIELD control. The following example shows how to use "code" as an additional value in a second FIELD or BMOBILE:FFIELD control:

```
<autocomplete id="mycity" source="file"
sourcelocation="./autocomplete/mycities.txt"></autocomplete>
<itr>
     <label name="Select a city"></label>
     <field valueprop="fldcity" autocompleteref="mycity"
      autocompletedisplayname="code" autocompletedisplayref="fldcode"></field>
     <hdist></hdist>
    <field valueprop="fldcode" displayonly="true"></field>
</itr>
```

The additional data will automatically be shown in the drop-down box.



The check for matching items is done on both "value" and "code". This means, when the user types "6", the list will be populated in the following way:



If you do not like this special handling for the additional data, choose the “Value and Label” format described below.

Value and Label

Example:

```
[
{ "label":"33729 Bielefeld", "value":"Bielefeld", "code":"33729" },
{ "label":"64397 Darmstadt", "value":"Darmstadt", "code":"64397" },
{ "label":"60329 Frankfurt", "value":"Frankfurt", "code":"60329" },
{ "label":"76135 Karlsruhe", "value":"Karlsruhe", "code":"76135" }
]
```

The "label" is shown in the drop-down box and the "value" is taken over to the FIELD or BMOBILE:FFIELD control. The check for matching items is done on the "label" as a whole. Typing "D" with `matchmode="start"` will result in no items.

Additional data can be specified in the FIELD or BMOBILE:FFIELD control using the properties `autocompleteref` and `autocompletedisplayname` as described above for the “Value” format.

Url

Choose this data source if the data is provided by a service. This is usually the case when the number of total values is very high and/or the values are read from databases like Adabas. This is the most flexible data source. Other than with the data sources String and File, the service is in charge of finding the matching values. The service will only return the values which match the characters typed in by the end-user. The service is free to implement any match mode it likes; therefore, the `matchmode` property is not supported.

The service must return the data in a simple JSON format. The same formats as described for the data source File are supported. The characters that the user types in are passed as request parameter "q" to the service.

Example: When the user types the characters "Da", the service *http://myhost/MyService* will be called as follows: *http://myhost/MyService?q=Da*.

A service can be implemented as a simple Java Servlet. With Natural RPC and EntireX, a service can also be written in Natural. The NJXDEMOS contain an example Natural subprogram, corresponding EntireX wrapper classes and configuration settings. See the document *NaturalAutocompleteRPCService.pdf* in the subfolder *njxdemos\autocomplete* of the NaturalAjaxDemos example project.

Properties

Basic			
id	The id of the autocomplete data source. This id can be referenced from FIELD controls.	Obligatory	
source	The kind of data source like string, file, url.	Optional	string file url
sourcelocation	The source location. Depends on the kind of source. Examples: ./autocomplete/myfile, http://myremotedatasource...	Optional	
minlength	The drop down selection box is not opened before a minlength number of characters is inserted into the FIELD control.	Optional	1 2 3 int-value
delay	The delay in milliseconds between when a keystroke occurs and when a search is performed	Optional	1

			2 3 int-value
matchmode	Specify "start" to find all items which start with the typed in characters. Specify "all" to find all items which contain the typed in characters. Default is "start". For source="url", the matchmode is ignored.	Optional	all start
rows	Height of the drop down box in rows. If more items are found, a scrollbar is shown.	Optional	1 2 3 int-value
maxresults	The maximum number of results displayed in the dropdown box.	Optional	1 2 3 int-value
Binding			
sourcelocationprop	Name of the adapter parameter that specifies the sourcelocation dynamically at runtime.	Optional	
sourceprop	Name of the adapter parameter that specifies the kind of source dynamically at runtime.	Optional	
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate	Optional	

	a generated statusprop variable to which field the statusprop belongs.		

92 TIMER

- Example 724
- Properties 725

With a timer, you can regularly trigger a defined event sent by the client. For example, you can use a timer to regularly update information to be displayed inside your page.

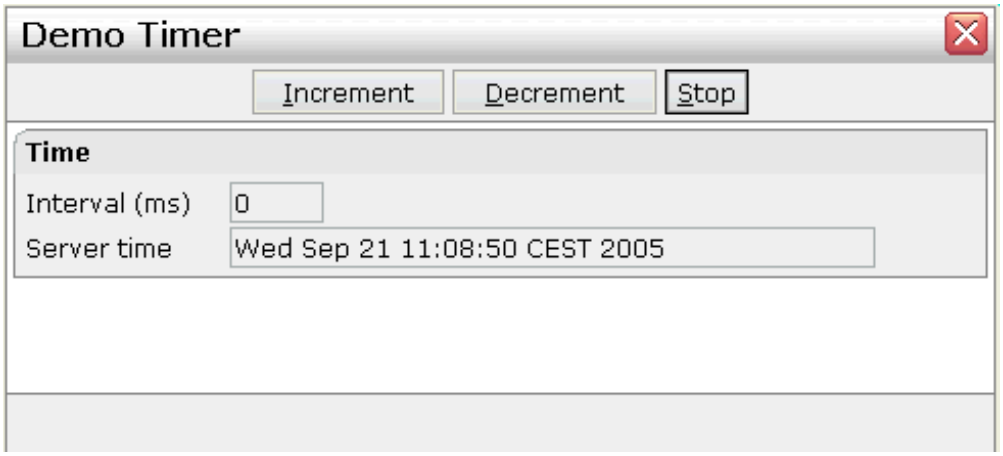
The timer tag is accessible as a valid subnode inside the page tag.

Specify either the `interval` or the `intervalprop` property in order to set the interval. In case of using a property for dynamically setting the interval, note the following:

- You can change the interval time at any time.
- You can stop the timer by setting the interval time to 0.

Example

The following screen displays a time stamp of the server. It is refreshed depending on the interval field. Increase/decrease the interval time by choosing the corresponding buttons.



The XML layout definition is:

```
<?xml version="1.0" encoding="UTF-8"?>
<natpage natsinglebyte="true" ↩
xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <titlebar name="Demo Timer">
    </titlebar>
    <header withdistance="false">
        <button name="~~Increment" method="incrementTimer">
        </button>
        <button name="~~Decrement" method="decrementTimer">
        </button>
        <button name="~~Stop" method="stopTimer">
        </button>
    </header>
    <pagebody>
```

```
        <rowarea name="Time">
            <itr>
                <label name="Interval (ms)" width="100" asplaintext="true">
                </label>
                <field valueprop="interval" length="5" displayonly="true" ↩
datatype="int">
                </field>
            </itr>
            <itr>
                <label name="Server time" width="100" asplaintext="true">
                </label>
                <field valueprop="serverTime" length="50" displayonly="true">
                </field>
            </itr>
        </rowarea>
    </pagebody>
    <statusbar withdistance="false">
    </statusbar>
    <timer intervalprop="interval">
    </timer>
</natpage>
```

In this example, the timer tag does not send a defined event but refreshes the screen. The timer interval is retrieved by the property `interval` of the adapter object.

Properties

Basic			
interval	Duration in milliseconds the timer waits between calling the adapter method defined in the METHOD property. Use this property to "hard code" the duration - or use INTERVALPROP to define the duration by an adapter property.	Sometimes obligatory	
intervalprop	Name of the adapter parameter that defines the timer interval duration. If 0 is passed then the timer is stopped.	Sometimes obligatory	
method	Name of the event that is sent to the adapter by the timer.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

93 XCIDATADEF - Data Definition

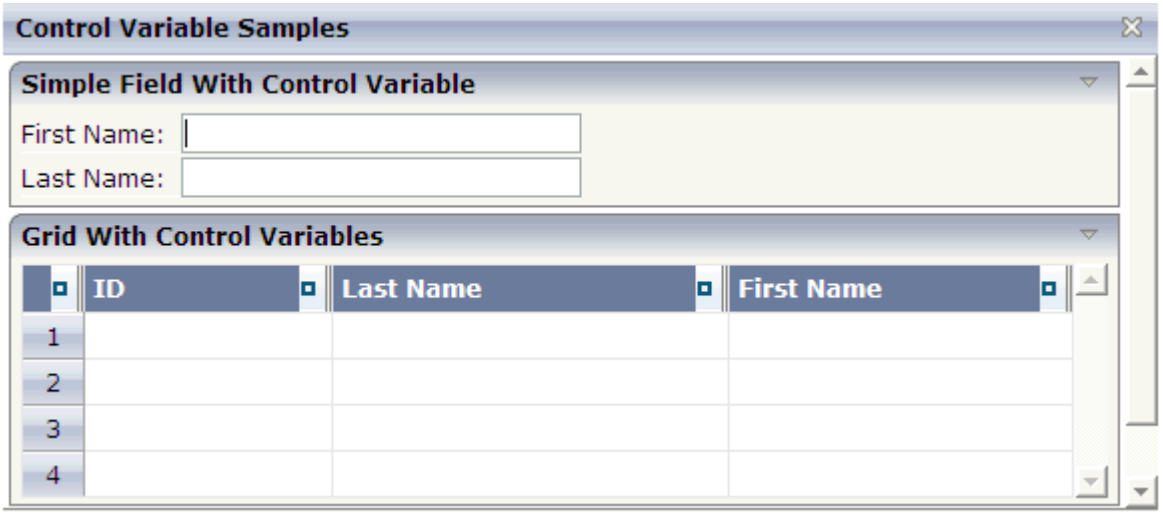
- Example 728
- Properties 731

With an XCIDATADEF control, you can define data structures that are exchanged between a page and its adapter, but which are not visually represented on the page. Examples are Natural control variables, which can be assigned to controls on a page after they have been defined in an XCIDATADEF control. They are not visually represented on the page, but can be evaluated by the application to control the modification status of the page and its controls.

The XCIDATADEF control allows the definition of scalar variables, structures, arrays, structures of arrays and arrays of structures. When an adapter is generated from a page that contains one or more XCIDATADEF controls, corresponding Natural data structures are generated into the parameter data area of the adapter.

Example

The following example shows several field controls and a grid control. It uses XCIDATADEF controls to define control variables and assigns these in various ways to the fields and grid elements of the page.



The XML layout definition is:

```
<?xml version="1.0" encoding="UTF-8"?>
<natpage hotkeys="13;onEnter" natsource="CTRCV-A" natsinglebyte="true" natcv="cv-page"
  xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <titlebar name="Control Variable Samples">
    </titlebar>
    <pagebody takefullheight="true">
        <rowarea name="Simple Field With Control Variable">
            <itr>
                <label name="First Name:" width="80">
                </label>
                <field valueprop="firstname" width="200" njx:natcv="cv-firstname">
```

```
            </field>
            <hdist width="50">
            </hdist>
        </itr>
        <itr>
            <label name="Last Name:" width="80">
            </label>
            <field valueprop="lastname" width="200" njx:natcv="cv-lastname">
            </field>
        </itr>
    </rowarea>
    <rowarea name="Grid With Control Variables">
        <rowtablearea2 griddataprop="persons" rowcount="4" width="100%">
            <tr>
                <gridcolheader width="30" propref="selected">
                </gridcolheader>
                <gridcolheader name="ID" width="25%" propref="id">
                </gridcolheader>
                <gridcolheader name="Last Name" width="40%" propref="last">
                </gridcolheader>
                <gridcolheader name="First Name" width="35%" propref="first">
                </gridcolheader>
            </tr>
            <repeat>
                <str valueprop="selected">
                    <selector valueprop="selected" singleselect="true">
                    </selector>
                    <field valueprop="id" width="25%" noborder="true"
                      transparentbackground="true">
                    </field>
                    <field valueprop="last" width="40%" noborder="true"
                   transparentbackground="true" njx:natcv="persons(*).cv-last">
                    </field>
                    <xcidatadef dataprop="cv-last" datatype="C">
                    </xcidatadef>
                    <field valueprop="first" width="35%" noborder="true"
                      transparentbackground="true" njx:natcv="cv-first(*)">
                    </field>
                </str>
            </repeat>
        </rowtablearea2>
    </rowarea>
    <rowarea name="Description" height="100%">
        <itr takefullwidth="true" height="100%">
            <subpage valueprop="infopagename" height="100%" width="100%">
            </subpage>
        </itr>
    </rowarea>
</pagebody>
<statusbar withdistance="false">
</statusbar>
<xcidatadef dataprop="cv-firstname" datatype="C">
```

```
    </xcidatadef>
    <xcidatadef dataprop="cv-lastname" datatype="C">
    </xcidatadef>
    <xcidatadef dataprop="cv-first" datatype="C" array="true">
    </xcidatadef>
</natpage>
```

The above example shows various ways in which control variables can be defined and assigned to controls:

- `cv-page` is a scalar control variable that is assigned to the page as a whole. In the application, it reflects the modification status of the entire page.
- `cv-firstname` and `cv-lastname` are scalar control variables that are assigned to the FIELD controls `firstname` and `lastname`. They reflect the modification status of the respective controls.
- `cv-last` is a control variable that is defined as an element of the grid `persons`. Consequently, it is implicitly an array and is assigned to the element `last` as `persons(*).cv-last`. Each occurrence of `persons(*).cv-last` reflects the modification status of the corresponding occurrence of `persons.last`.
- `cv-first` is an array of control variables that is defined outside the grid `persons`. Consequently, it is assigned to the element `first` as `cv-first(*)`. Each occurrence of `cv-first(*)` reflects the modification status of the corresponding occurrence of `persons.first`. Note the difference to the previous case: because `cv-first(*)` is defined outside the grid `persons`, it is not automatically resized together with `persons.first`. Resizing `cv-first(*)` appropriately is in the responsibility of the application program.

The corresponding adapter code looks as follows:

```
DEFINE DATA PARAMETER
/*( PARAMETER
1 CV-FIRST (C/1:*)
1 CV-FIRSTNAME (C)
1 CV-LASTNAME (C)
1 CV-PAGE (C)
1 FIRSTNAME (A) DYNAMIC
1 INFOPAGENAME (A) DYNAMIC
1 LASTNAME (A) DYNAMIC
1 PERSONS (1:*)
2 CV-LAST (C)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 SELECTED (L)
/*) END-PARAMETER
END-DEFINE
...
/*( PROCESS PAGE
PROCESS PAGE (CV=CV-PAGE) U'/njxdemos/ctrlcontrolvar' WITH
PARAMETERS
```

```
 NAME U'firstname'
  VALUE FIRSTNAME (CV=CV-FIRSTNAME)
 NAME U'infopagename'
  VALUE INFOPAGENAME
 NAME U'lastname'
  VALUE LASTNAME (CV=CV-LASTNAME)
 NAME U'persons(*).first'
  VALUE PERSONS.FIRST(*) (CV=CV-FIRST(*))
 NAME U'persons(*).id'
  VALUE PERSONS.ID(*)
 NAME U'persons(*).last'
  VALUE PERSONS.LAST(*) (CV=PERSONS.CV-LAST(*))
 NAME U'persons(*).selected'
  VALUE PERSONS.SELECTED(*) (EM='false'/'true')
END-PARAMETERS
/*) END-PROCESS
...
```

The example code is contained in the Natural for Ajax demos as program `CTRCV-P`.

Properties

Basic			
datatype	Data type of the data element. One of the list of valid values. Using the reserved word "type" you can nest multiple XCIDATADEF structures.	Optional	type xs:string xs:int xs:float xs:decimal xs:double xs:date xs:dateTime xs:time xs:byte xs:short xs:boolean
array	If set to true, the XCIDATADEF will be an array.	Optional	true

			false
arraydimension	If ARRAY is set to TRUE you can specify the array dimension here. Supported values are "1" and "2". The default is "1".	Optional	1 2
clientdata	Default is false. If set to true then the data is also send to the browser. Usually applications use the default setting. Only set this to true for small data that is really rendered in the browser.	Optional	true false
Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	Optional	
njx:natsysio	If the control shall be bound to a Natural system variable with the attribute njx:natsysvar, this attribute indicates if the system variable is modifiable. The default is false.	Optional	
njx:natstringtype	If the control shall be bound to a Natural system variable of string format with the attribute njx:natsysvar, this attribute indicates the format of the string, A (code page) or U (Unicode). The default is A.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

94 XCICONTEXT

- General Information 734
- Example 734
- Properties 735

The XCICONTEXT control can be used to exchange simple data values between different pages of an application. Example usage scenarios are:

- Share simple data between a page and an embedded page (see also **SUBCISPAGE2** control). The pages can share, for instance, a key value for the currently selected article. The embedded page shows details based on this key.
- Workplace scenarios in which multiple pages need to share some simple data.

General Information

The XCICONTEXT control can hold multiple XCICONTEXTPARAM controls. Each of the XCICONTEXTPARAM controls defines a name and a value. At runtime, these names and values are automatically shared between all pages running in the same session.

The Natural applications change and access the corresponding data fields in the usual way. No additional coding is required for the data exchange - this is automatically done by the framework. Internally, the framework uses the session context (for more information on the session context, see the section *Saving Context Data* in the Application Designer documentation; this can be found under *Special Development Topics* > *Details on Session Management*).

Example

The example showing the SUBCISPAGE2 control in the Natural for Ajax demos also shows how to use the XCICONTEXT control. In the layouts of all pages that are to share a specific data value, a corresponding XCICONTEXTPARAM with the same name has been defined. In the example below, the pages share the value for the "selectedArticle":

```
<xcicontext contextprop="mycontext">
  <xcicontextparam valueprop="selectedArticle" lookupname="selectedArticle">
  </xcicontextparam>
</xcicontext>
```

The generated data Natural data structure looks as follows:

```
1 MYCONTEXT
2 SELECTEDARTICLE (A) DYNAMIC
```

The Natural applications can now access the `SELECTEDARTICLE` field in the usual way.

Properties

Natural			
njx:natname	If a Natural variable with a name not valid for Application Designer (for instance #FIELD1) shall be bound to the control, a different name (for instance HFIELD1) can be bound instead. If the original name (in this case #FIELD1) is then specified in this attribute, the original name is generated into the parameter data area of the Natural adapter and a mapping between the two names is generated into the PROCESS PAGE statement of the Natural adapter. This mapping must not break a once defined group structure. If for instance a grid control that is bound to a name of GRID1 contains fields that are bound to FIELD1 and FIELD2 respectively, the corresponding njx:natname values may be #GRID1.#FIELD1 and #GRID1.#FIELD2, but not #GRID1.#FIELD1 and #MYGRID1.#FIELD2.	Optional	
njx:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter, before the field name. The Map Converter, for instance, uses this attributes to indicate a generated statusprop variable to which field the statusprop belongs.	Optional	

95 NJX:XCIOPENPOPUP

- Example 738
- Adapter Interface 738
- Non-responsive pages 740
- Responsive pages 740

The NJX:XCIOPENPOPUP control is used to configure certain parameters of a pop-up dialog before it is opened with a `PROCESS PAGE MODAL` statement. The control does not have design-time properties, nor does it raise events.

For non-responsive pages, two types of pop-up dialogs are supported: browser pop-ups and page pop-ups. Browser pop-ups are controlled by the web browser. To use them, pop-ups must be enabled in the browser settings. Page pop-ups are rendered by Natural for Ajax. To use them, pop-ups need not be enabled in the browser settings.

For responsive pages, dynamic modals (BMOBILE:DYNMODAL) are supported. They are rendered as pure html therefore Browser pop-ups don't need to be enabled.

This control is intended to be used in the page that opens the pop-up dialog. Examples for the usage of this control are provided in the Natural for Ajax demos.

Example

The XML code for the example looks as follows:

```
<natpage xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <njx:xciopenpopup>
    </njx:xciopenpopup >
</natpage>
```

Adapter Interface

```
1 XCIOPENPOPUP
  2 FEATURES (A) DYNAMIC
  2 HEIGHT (I4)
  2 LEFT (I4)
  2 POPUPTYPE (A) DYNAMIC
  2 TITLE (A) DYNAMIC
  2 TOP (I4)
  2 WIDTH (I4)
```

or

```
1 XCIOPENPOPUP
  2 FEATURES (A) DYNAMIC
  2 HEIGHT (I4)
  2 LEFT (I4)
  2 POPUPTYPE (A) DYNAMIC
  2 TITLE (A) DYNAMIC
  2 TOPPOS (I4)
  2 WIDTH (I4)
```

`TOPPOS` is generated in the adapter interface when the property `natkcheck` in the **NATPAGE** control is set to "true", otherwise `TOP` is generated.

Each element of the structure controls a parameter of the next pop-up dialog to be opened.

Element	Meaning
`FEATURES`	Allows defining specific features of the pop-up dialog. See the property `popupfeatures` of the **NATPAGE** control.
`HEIGHT`	The height of the pop-up dialog. If you specify a value greater than 0, you also need to specify a value greater than 0 for `WIDTH`.
`LEFT`	The left position of the pop-up dialog. If you specify a value greater than 0, you also need to specify values greater than 0 for `HEIGHT` and `WIDTH`.
`POPUPTYPE`	Specifies whether the pop-up dialog is to be opened as a browser pop-up (value "POPUP"), as a page pop-up (value "PAGEPOPUP"), as a responsive pop-up in a responsive page (value "BMPOPUP"), or as a responsive sliding panel in a responsive page (value "BMPANEL").
`TITLE`	The title to be displayed in the caption of the pop-up dialog.
`TOP/TOPPOS`	The top position of the pop-up dialog. For non-responsive pages: if you specify a value greater than 0, you also need to specify values greater than 0 for `HEIGHT` and `WIDTH`.
`WIDTH`	The width of the pop-up dialog. For non-responsive pages: if you specify a value greater than 0, you also need to specify a value greater than 0 for `HEIGHT`.

Valid values for `HEIGHT`, `WIDTH`, `LEFT` and `TOP/TOPPOS` are -1, 0 or a value greater than 0. The value -150, for example, is not a valid value.

Special settings:

- To open a centered pop-up, set `LEFT` and `TOP/TOPPOS` to -1.
- To open a pop-up in the upper left corner of the browser output area, set `LEFT` and `TOP/TOPPOS` to 0.
- To open a pop-up with the default settings, set `HEIGHT`, `WIDTH`, `LEFT` and `TOP/TOPPOS` to -1.

Non-responsive pages

Valid values for HEIGHT, WIDTH, LEFT and TOP/TOPPOS are -1, 0 or a value greater than 0. The value -150, for example, is not a valid value.

Special settings:

- To open a centered pop-up, set LEFT and TOP/TOPPOS to -1.
- To open a pop-up in the upper left corner of the browser output area, set LEFT and TOP/TOPPOS to 0.
- To open a pop-up with the default settings, set HEIGHT, WIDTH, LEFT and TOP/TOPPOS to -1.

Responsive pages

HEIGHT, WIDTH, LEFT, TOP/TOPPOS values must be greater than 0. If the pop-up is bigger than the containing window, the values are automatically adapted so that the pop-up fits into the parent window. For supported features, see also the chapter “Dynamic Pop-Ups” in the Responsive Controls chapter.

96 NJX:XCILIVINGPOPUP

- Example 742
- Adapter Interface 742

The NJX:XCILIVINGPOPUP control is used to configure certain parameters of a pop-up dialog from inside the running pop-up dialog. The control does not have design-time properties, nor does it raise events.

Two types of pop-up dialogs are supported, browser pop-ups and page pop-ups. Browser pop-ups are controlled by the web browser. To use them, pop-ups must be enabled in the browser settings. Page pop-ups are rendered by Natural for Ajax. To use them, pop-ups need not be enabled in the browser settings.

This control is intended to be used in the page that is opened as a pop-up dialog. An example for the usage of the control is provided in the Natural for Ajax demos as program `CTRPOP-P`.

The following topics are covered below:

Example

The XML code for the example looks as follows:

```
<natpage xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <njx:xcilivingpopup>
    </njx:xcilivingpopup >
</natpage>
```

Adapter Interface

```
1 XCILIVINGPOPUP
  2 HEIGHT (I4)
  2 TITLE (A) DYNAMIC
  2 WIDTH (I4)
```

Each element of the structure controls a parameter of the next pop-up dialog to be opened.

Element	Meaning
`HEIGHT`	The height of the pop-up dialog. If you specify a value greater than 0, you also need to specify a value greater than 0 for `WIDTH`.
`TITLE`	The title to be displayed in the caption of the pop-up dialog.
`WIDTH`	The height of the pop-up dialog. If you specify a value greater than 0, you also need to specify a value greater than 0 for `HEIGHT`.

To open a pop-up with the default settings, set `HEIGHT` and `WIDTH` to -1.

97 Extended Hot Key Management

- Direct Hot Key Definitions with Certain Controls 744
- Hot Key Definitions for Certain Controls 744

Extended hot key management provides the following features:

- Possibility to define hot keys with certain controls.
- Possibility to define language dependent hot keys.

Direct Hot Key Definitions with Certain Controls

Some controls allow to directly specify hot keys within the text that is displayed inside the control. The controls that currently support this feature are:

- BUTTON
- MENU
- ROWTABAREA

Example: If you specify the button text to be "~~Stop", the button will look like this:



The text may both be directly maintained in the control (`name` property) or may come from the multi language management (`textid` property).

At the time, the hot key CTRL+ALT+S will be added to the page. The definition of hot keys in the texts of MENU controls or ROWTABAREA controls is done in the same way.

Caution: Application Designer does not check if hot keys are defined twice in a page.

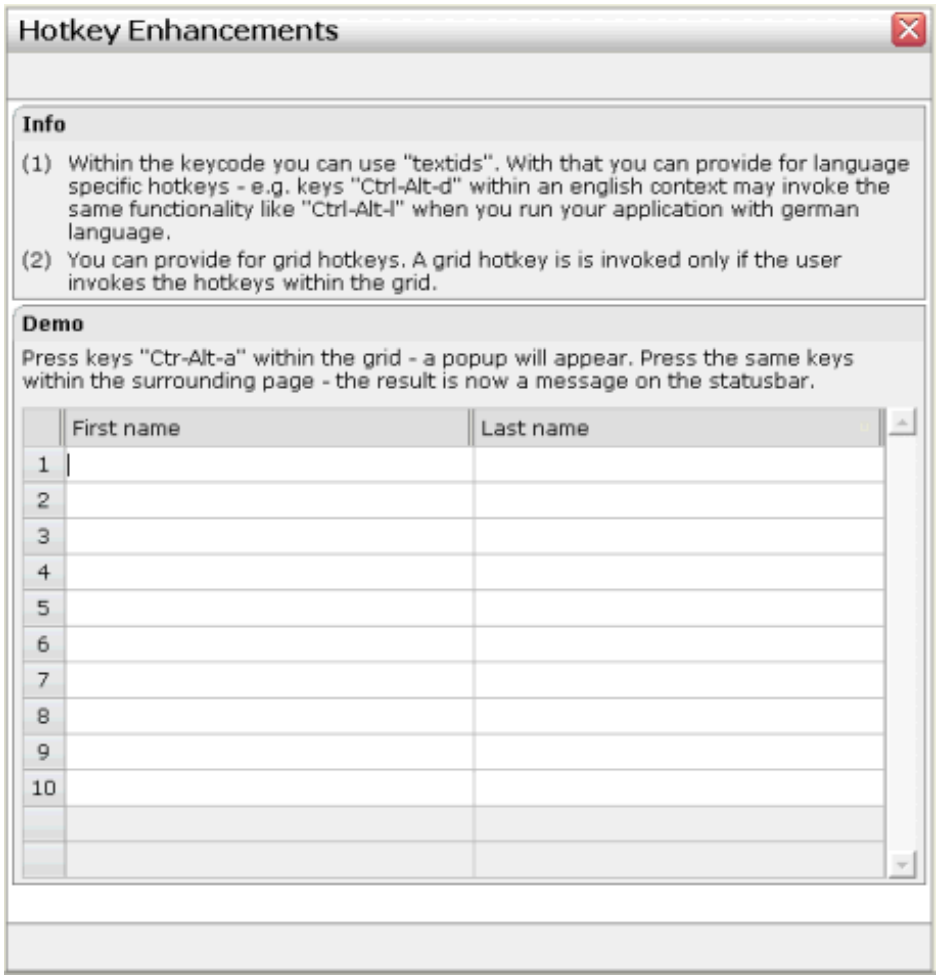
Why use CTRL+ALT as a default way to trigger the hot keys? This is because most of the simple ALT keys are already occupied by the browser.

Hot Key Definitions for Certain Controls

The controls PAGE, FIELD and ROWTABLEAREA2 support the property `hotkeys`.

The `hotkeys` property defines the active hot keys for the corresponding control. This means that you may have hot keys that are only valid inside a certain grid (ROWTABLEAREA2 control) or even inside a single FIELD, but are not valid inside the whole page (PAGE control).

Have a look at the following demo:



If the user presses CTRL+ALT+A inside the grid, the hot key is managed by the grid. If the user presses the same key outside the grid, the hot key is processed by a corresponding definition on page level. The XML layout looks as follows:

```
<page model="com.softwareag.cis.test40.GridHotkeysAdapter" ↩
translationreference="40_gridhotkeys"
      hotkeys="ctrl-alt-65;onCtrlAltAPage">
...
...
        <rowtablearea2 griddataprop="grid" rowcount="12" width="100%" ↩
firstrowcolwidths="true"
                      hotkeys="ctrl-alt-$KEYCODE_A;onCtrlAltA">
...
...
```

The `hotkeys` property on PAGE, FIELD or ROWTABLEAREA2 is a semicolon-separated list containing the hot key itself and the method it is calling. There can be multiple hot key definitions for the same control. When maintaining this property, use the special dialog in the Layout Painter that appears for the `hotkeys` property. For further information, see *Defining Hot Keys* in the *Development Workplace* documentation.

You can either specify the key code of the hot key or a text ID that is to be translated by the multi language management.

98 Function Key Handling

Some keyboard function keys are usually assigned to specific functions of the web browser. F5, for example, causes a page reload and F11 toggles full screen mode.

In a Natural for Ajax application, these keyboard function keys might be assigned as hot keys to events in the application. But the user should also have the option to use, for example, F11 in the usual way as a web browser function key. Therefore, the following rules apply:

- If the keyboard focus is on the Natural for Ajax page, the function key raises the corresponding event in the application.
- If the keyboard focus is not on the Natural for Ajax page, but in the area of the web browser (for example, in the address line), the function key raises the corresponding event in the web browser.

99 NJX:OBJECTS

- General Information 750
- Example 751
- Adapter Interface 751

The NJX:OBJECTS control is used to make assets that are contained in Natural binary variables (BLOBs) available to controls on the web page. Examples are images in arbitrary controls such ICON, IMAGEOUT, CLIENTTREE or MENU.

General Information

In the Natural program, the NJX:OBJECTS control is represented with the following data structure:

```
1 XCIOBJECTS (1:*)
2 CONTENT (B) DYNAMIC
2 CONTENTID (A) DYNAMIC
2 CONTENTTYPE (A) DYNAMIC
```

With this data structure, images and other binary content can be transported from the Natural program to the application server or web container.

For each entry in the data structure, the following must be specified: the binary content itself, an identifier and (optionally) the content type. In your Natural program, you only add the binary content once, with a specific name. The binary content is then automatically cached in the application server or web container until the Natural application completes its execution. During a server roundtrip from Natural to the application server or web container, all binary content that is found for this data structure is added to the cache and the data structure is cleared. After a server roundtrip, the data structure is always empty.

When referencing binary content such as an image from within a control of your page layout, you usually specify an URL. For Natural binary content, the URL starts with the prefix "nat:" followed by a name. To reference binary content from within a design-time property such `image` in a control of your page layout, you specify this URL directly in your page layout. For example:

```
...
<icon image="nat:icon01">
</icon>
...
```

When referencing binary content using a dynamic property such as `imageprop`, you can use automatically generated names.

In both cases (when you apply the URLs to properties statically at design-time and when you apply the URLs dynamically at runtime), you do not have to work directly on the `XCIOBJECTS` structure. Instead, you can use the helper subprogram `MAKEURL` which can be found in the Natural for Ajax demos. You can use your own names or you can leave it to the helper subprogram `MAKEURL` to generate the names. The `MAKEURL` subprogram adds a corresponding entry to the `XCIOBJECTS` data structure and returns an URL for this binary content. When the `CALLNAT` statement in the example below has been executed, the `MYURL` field contains a valid URL which you can apply to dynamic image properties (for example, in a grid or other dynamic controls).

```
DEFINE DATA LOCAL
1 MYURL (A) DYNAMIC
1 XCIOBJECTS (1:*)
2 CONTENT (B) DYNAMIC
2 CONTENTID (A) DYNAMIC
2 CONTENTTYPE (A) DYNAMIC
...
LOCAL
1 MYBLOB (B) DYNAMIC
...
END-DEFINE

CALLNAT "MAKEURL" XCIOBJECTS(*) MYBLOB MYURL
```

If you want to use your own name (for example, "icon01"), you have to specify the `CALLNAT` statement as follows:

```
CALLNAT "MAKEURL" XCIOBJECTS(*) MYBLOB MYURL "icon01"
```

Optionally, you can explicitly specify the content type. This is required for content types which are not automatically recognized by the browser. Example:

```
CALLNAT "MAKEURL" XCIOBJECTS(*) MYBLOB MYURL "icon01" "gif"
```

Note: See also ***Images*** in *Some Common Rules for all Controls*.

Example

Examples which show the usage of the NJX:OBJECTS control are provided in the Natural for Ajax demos: the programs `CTROB1-P` and `CTROB2-P`.

Adapter Interface

```
1 XCIOBJECTS (1:*)
2 CONTENT (B) DYNAMIC
2 CONTENTID (A) DYNAMIC
2 CONTENTTYPE (A) DYNAMIC
```

Element	Description
CONTENT	The binary content as contained in a Natural binary variable.
CONTENTID	A name used for caching in the application server or web container and for referencing the content from within an URL.
CONTENTTYPE	The content type as understood by a browser. For content types which are not automatically recognized by a browser, the content type has to be specified.

100 NJX:SESSIONPARAMS

- General Information 754
- Example 754
- Adapter Interface 755

The NJX:SESSIONPARAMS control is used to modify the following Natural for Ajax session parameters in the Natural application:

- `STYLE`: By specifying this parameter in the Natural application, you can change the style sheet of a running Natural for Ajax session.
- `FIRSTDAYINWEEK`: By specifying this parameter in the Natural application, you can define either Sunday or Monday as the first day in the week.

General Information

In the Natural program, the NJX:SESSIONPARAMS control is represented with the following data structure:

```
1 XCISESSIONPARAMS
2 FIRSTDAYINWEEK (U) DYNAMIC
2 STYLE (U) DYNAMIC
```

Possible values for the `FIRSTDAYINWEEK` parameter are "SU" for Sunday and "MO" for Monday. Other values are not supported.

The `STYLE` parameter must contain the name of the style sheet that is to be used, including the path. For example, "../cis/styles/CIS_DEFAULT.css".

The following sample code shows how to change the Natural for Ajax session parameters in the Natural application:

```
XCISESSIONPARAMS.STYLE := '../cis/styles/CIS_DEFAULT.css'
XCISESSIONPARAMS.FIRSTDAYINWEEK := 'MO'   /* 'SU' or 'MO'
. . .
PROCESS PAGE UPDATE FULL
```

Example

An example program `CTRSEP-P` which shows the usage of the NJX:SESSIONPARAMS control is provided in the Natural for Ajax demos.

Adapter Interface

```
1 XCISESSIONPARAMS
2 FIRSTDAYINWEEK (U) DYNAMIC
2 STYLE (U) DYNAMIC
```

Element	Description
STYLE	The name of the style sheet.
FIRSTDAYINWEEK	The first day of the week. This can be either "SU" for Sunday or "MO" for Monday.

101 NJX:REQUESTCONTEXT

- General Information 758
- Adapter Interface 760

With the NJX:REQUESTCONTEXT control, the Natural application can access context information regarding the request:

- Version information regarding the Natural for Ajax and Application Designer runtime.
- Client information regarding the web client which is sending the request.
- Server information regarding the web server which is executing the request.
- Session information regarding the executed request.

General Information

The complete generated data structure is shown below, see *Adapter Interface*.

Version Information

The following fields provide version information:

Field	Description
NJXVERSION	The version number of the Natural for Ajax runtime which is executing the request. Example: 8.3.3.1
CISVERSION	The unique version number of the Application Designer component which is contained in the Natural for Ajax product. The value is read from the *cisversion.xml* file of the Natural for Ajax web application. Example: CIS_V833_20140110_1943_NJX

Client Information

The following fields provide client information:

Field	Description
CLIENTIPADDR	Contains the value of `getRemoteAddr()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getRemoteAddr()*. Example: 203.0.113.195
CLIENTHOSTNAME	Contains the value of `getRemoteHost()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getRemoteHost()*. Example: MYCLIENT **Note:** To see the real host name, you might have to change some configuration settings in your application server. For instance, Tomcat only shows the real host name if

Field	Description
	`enableLookups` is set to "true". For more information, see the corresponding documentation of your application server.
CLIENTPORT	Contains the value of `getRemotePort()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getRemotePort()*. Example: 65021
CLIENTLOCALE	Contains a string representation (`toString()`) of the object returned by `getLocale()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getLocale()*. Example: en_US
CLIENTLOCALES	Contains string representations (`toString()`) for the objects returned by `getLocales()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getLocales()*. Example: en_US en

Server Information

The following fields provide server information:

Field	Description
SERVERHOSTNAME	Contains the value of `getServerName()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getServerName()*. Example: MYHOST
SERVERPORT	Contains the value of `getServerPort()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/ServletRequest.html#getServerPort()*. Example: 8080

Session Information

The following fields provide session information:

Field	Description
HTTPSESSIONID	Contains the value of `getId()` for the corresponding HTTP servlet request. See also *https://docs.oracle.com/javaee/6/api/javax/servlet/http/HttpSession.html#getId()*. Example: 2DE02F9A39E3AAA498D964C5CB655FD3
CISSESSIONID	This is a unique ID inside one Natural for Ajax web application. The ID contains the Application Designer session ID and subsession ID (see also the information on session IDs and on HTTP sessions in the Application Designer documentation - both topics are contained in *Special Development Topics*). Inside one Natural for Ajax web application,

Field	Description
	the value of CISSESSIONID uniquely identifies the corresponding Natural connection for this request. Example: CASA2_184904389529_138947393109741706295495795534l8

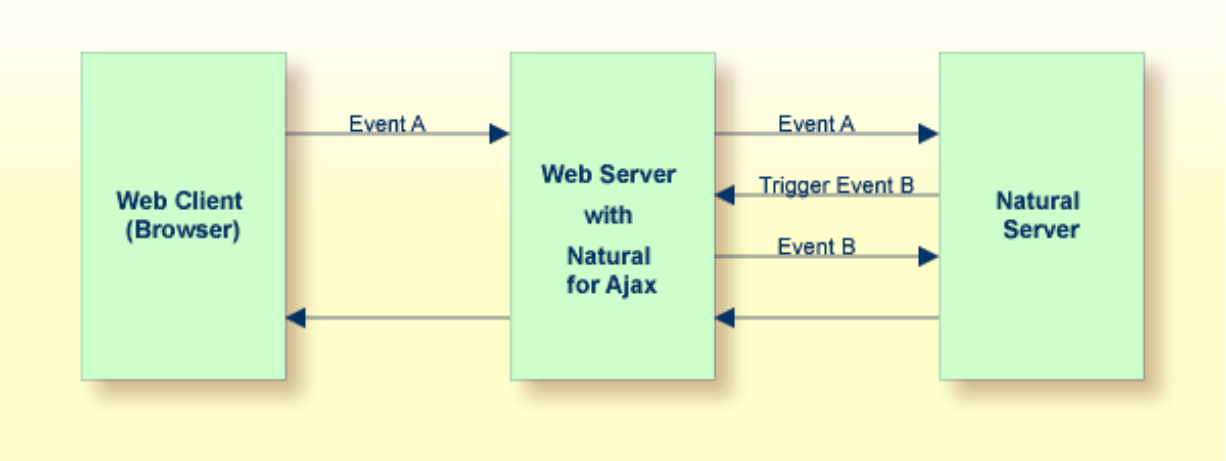
Adapter Interface

```
1 XCIREQUESTCONTEXT
2 CISSESSIONID (A) DYNAMIC
2 CISVERSION (A) DYNAMIC
2 CLIENTHOSTNAME (A) DYNAMIC
2 CLIENTIPADDR (A) DYNAMIC
2 CLIENTLOCALE (A) DYNAMIC
2 CLIENTLOCALES (A/1:*) DYNAMIC
2 CLIENTPORT (I4)
2 HTTPSESSIONID (A) DYNAMIC
2 NJXVERSION (A) DYNAMIC
2 SERVERHOSTNAME (A) DYNAMIC
2 SERVERPORT (I4)
```

102 NJX:TRIGGEREVENT

- Examples 762
- Adapter Interface 764

The NJX:TRIGGEREVENT control can be used to trigger a server roundtrip between the web application and the Natural server.



Examples

Common usage scenarios for the NJX:TRIGGEREVENT control are scenarios in which the event reaction of your Natural application consists of several parts which contain data handling and page navigation or pop-up handling. Triggering an event allows you to modularize and combine these different event reaction parts. Some specific sample use cases are described below.

Pop-ups

Your application might want to navigate to a specific page when a pop-up is closed. The navigation may depend on some return value of the pop-up. When the pop-up is closed, the Natural application can trigger an event depending on the pop-up result. In the triggered event, the Natural application can then do the navigation. The following code shows how this can be done:

```
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end',U'nat:browser.end'
  /* Page closed.
  IGNORE
 VALUE U'onReactionA'
  FETCH RETURN 'MYPRGA'
  PROCESS PAGE UPDATE FULL
 VALUE U'onReactionB'
  FETCH RETURN 'MYPRGB'
  PROCESS PAGE UPDATE FULL
 VALUE U'onOpenMyPopup'
  PROCESS PAGE MODAL
   CALLNAT 'MYPOPUP' MYRETURN
  END-PROCESS
```

```
  IF MYRETURN = 'A' THEN
     XCITRIGGEREVENT:='onReactionA'
  END-IF
  IF MYRETURN = 'B' THEN
     XCITRIGGEREVENT:='onReactionB'
  END-IF
  PROCESS PAGE UPDATE FULL
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
```

Workplace Applications

In a workplace application, you might want to open a logon page as the first page. On logon, your Natural application would dynamically define the function tree of the workplace and after that the logon page should be closed. The following code shows how this can be done:

```
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end',U'nat:browser.end'
  /* Page closed.
  IGNORE
 VALUE U'onReady'
  /* Close the Login page.
  RESIZE ARRAY XCIWPACCESS2 TO (1:1)
  CMDCLOSECONTENTPAGE(1) := 'closeit'
  PROCESS PAGE UPDATE FULL
 VALUE U'onLogin'
  /* Define the function tree of the
  /* workplace dynamically
  ...
  ...
  /* Trigger the event 'onReady'.
  XCITRIGGEREVENT:='onReady'
  PROCESS PAGE UPDATE FULL
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
```

TIMER Control

Using the **TIMER** control, you can automatically trigger an event repeatedly, based on the time interval. When navigating to a different page or when opening a page pop-up, you would like to stop the timer and start it again when returning to the page. This stopping and restarting of the timer can be achieved using an NJX:TRIGGEREVENT control. The following code shows how to stop the timer before opening a page pop-up, and how to restart the timer after closing the page pop-up.

```
DECIDE ON FIRST *PAGE-EVENT
 VALUE U'nat:page.end',U'nat:browser.end'
  /* Page closed.
  IGNORE
 VALUE U'onOpenMyPopup'
  PROCESS PAGE MODAL
   CALLNAT 'MYPOPUP'
  END-PROCESS
  /* start the timer again
  MYTIMERINTERVAL:=5000
  PROCESS PAGE UPDATE FULL
 VALUE U'onShowDetails'
  /* Stop the timer
  MYTIMERINTERVAL:=0
  /* customize popup settings
  XCIOPENPOPUP.POPUPTYPE:='PAGEPOPUP'
  XCIOPENPOPUP.TITLE:= 'Show Details'
  ...
  /* trigger an event to open the popup
  XCITRIGGEREVENT:='onOpenMyPopup'
  PROCESS PAGE UPDATE FULL
 NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
```

Adapter Interface

```
1 XCITRIGGEREVENT (A) DYNAMIC
```

You have to specify the name of the event to be triggered.

VIII Working with Pop-Ups

103 Working with Pop-Ups

- Pop-Ups - Non-Responsive Pages 768
- Pop-Ups - Responsive Pages 772

This part deals with applications that use pop-ups to display content.

The following topics are covered below:

Pop-Ups - Non-Responsive Pages

- Basics
- Browser Pop-Up Support
- Using Page Pop-Ups
- Special Considerations when using Internet Explorer 11

Basics

You can open normal Natural pages as pop-ups. Use the PROCESS PAGE MODAL in your Natural program (see *Using Pop-Up Windows*). Per default, all pop-ups are opened as page pop-ups.

To define how to open pop-ups, use one or both of the following:

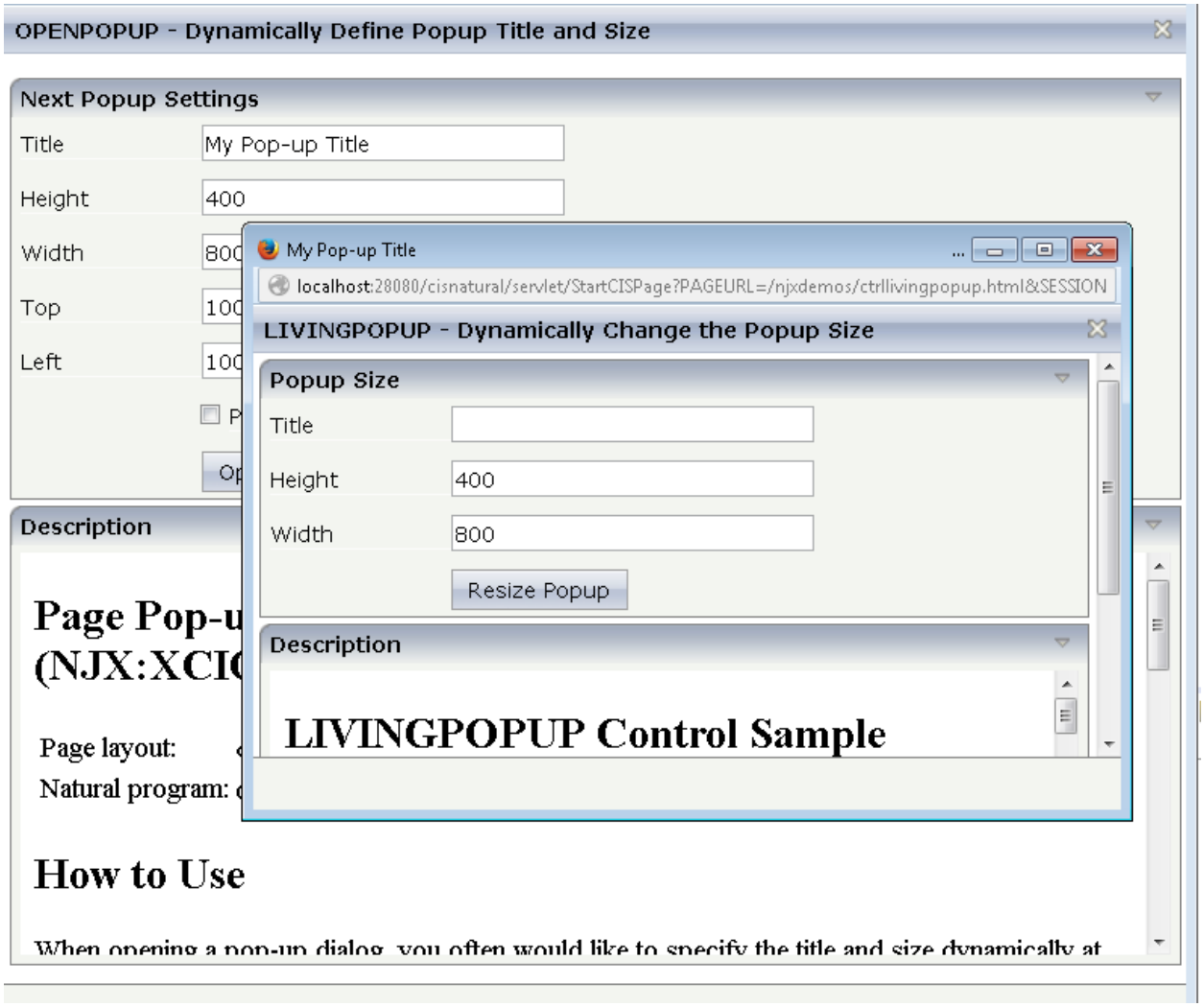
- The properties `popupheight`, `popupwidth` and `popupfeature` in the ***NATPAGE*** control of the pop-up layout.
- The control **NJX:XCIOPENPOPUP** in the parent layout.

If both are used, the settings in the NJX:XCIOPENPOPUP data structures of the Natural program will overwrite the settings in the NATPAGE control.

Browser Pop-Up Support

Some time ago, browsers easily supported real modal pop-ups. Pop-ups could be configured so that the address bar was not visible. Meanwhile, the recommendation in browsers is to activate pop-up blockers. And even when explicitly allowing browser pop-ups, the Natural for Ajax browser pop-ups are no longer real modal pop-ups: they can be minimized and look different.

The reason is that modal browser pop-ups are regarded as security risks. The browsers either do not support real modal browser pop-ups anymore or only when the security settings in the browser are changed.



Using Page Pop-Ups

There is a solution to the above-mentioned problem: page pop-ups. Natural for Ajax also supports this different type of pop-up. Page pop-ups are real modal pop-ups and do not increase security risks. Because they are not classic browser pop-ups, a browser´s pop-up blocker can be activated and the page pop-ups will still work correctly.

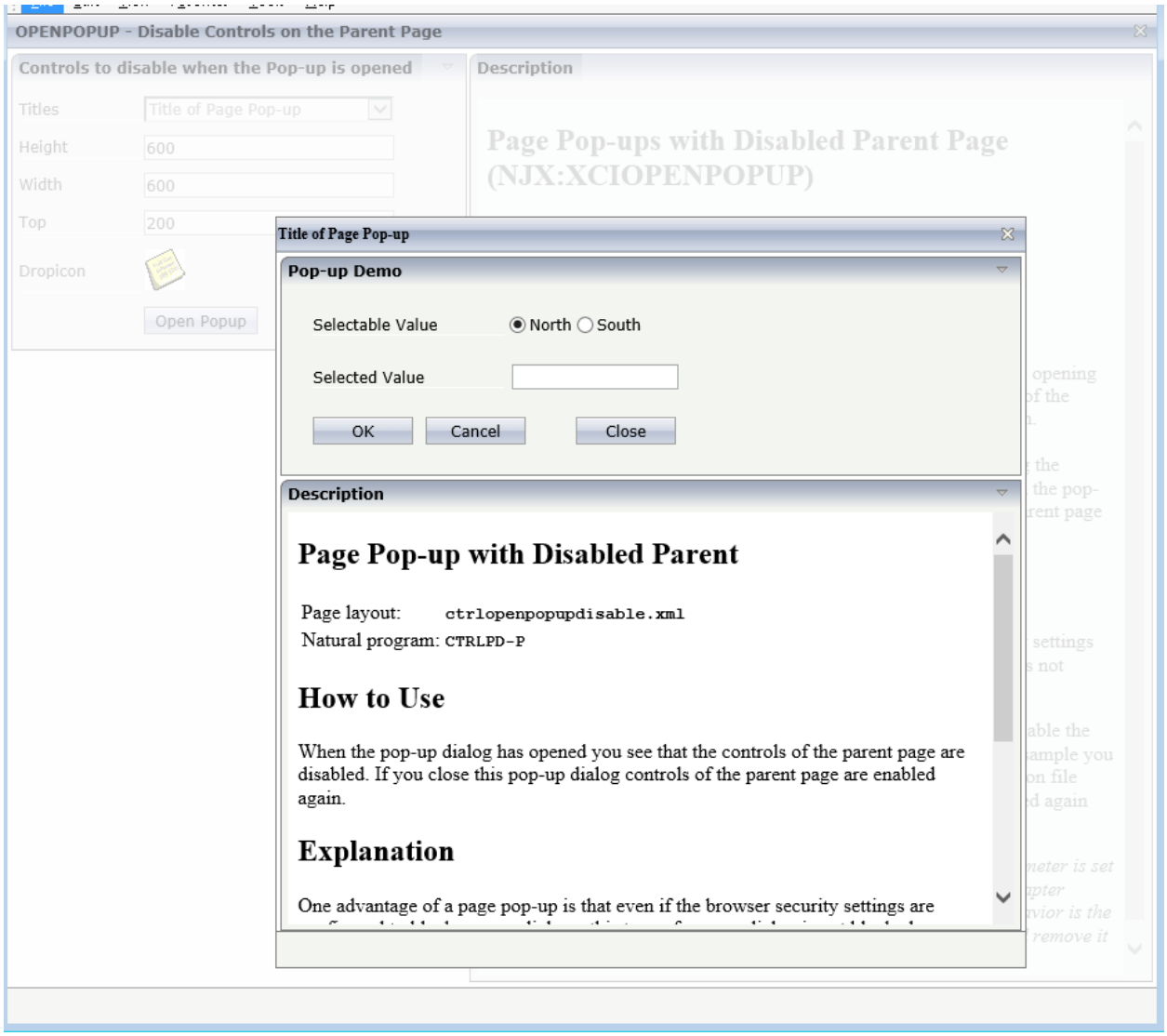
In previous versions of Natural for Ajax, switching to page pop-ups required source changes in the Natural for Ajax applications using them. This is no longer necessary. There is a configuration setting which allows opening the Natural for Ajax application pop-ups as page pop-ups:

- Open the file *cisconfig.xml* in a text or XML editor.

Tip: NaturalONE supports adding custom *cisconfig.xml* files to User Interface components and to automatically package them during deployment.

- Set the attribute `usepagepopup="true"`

For page pop-ups you can also configure in the *cisconfig.xml* to disable the parent page while the page pop-up is open. The NaturalAjaxDemos contain a corresponding sample:

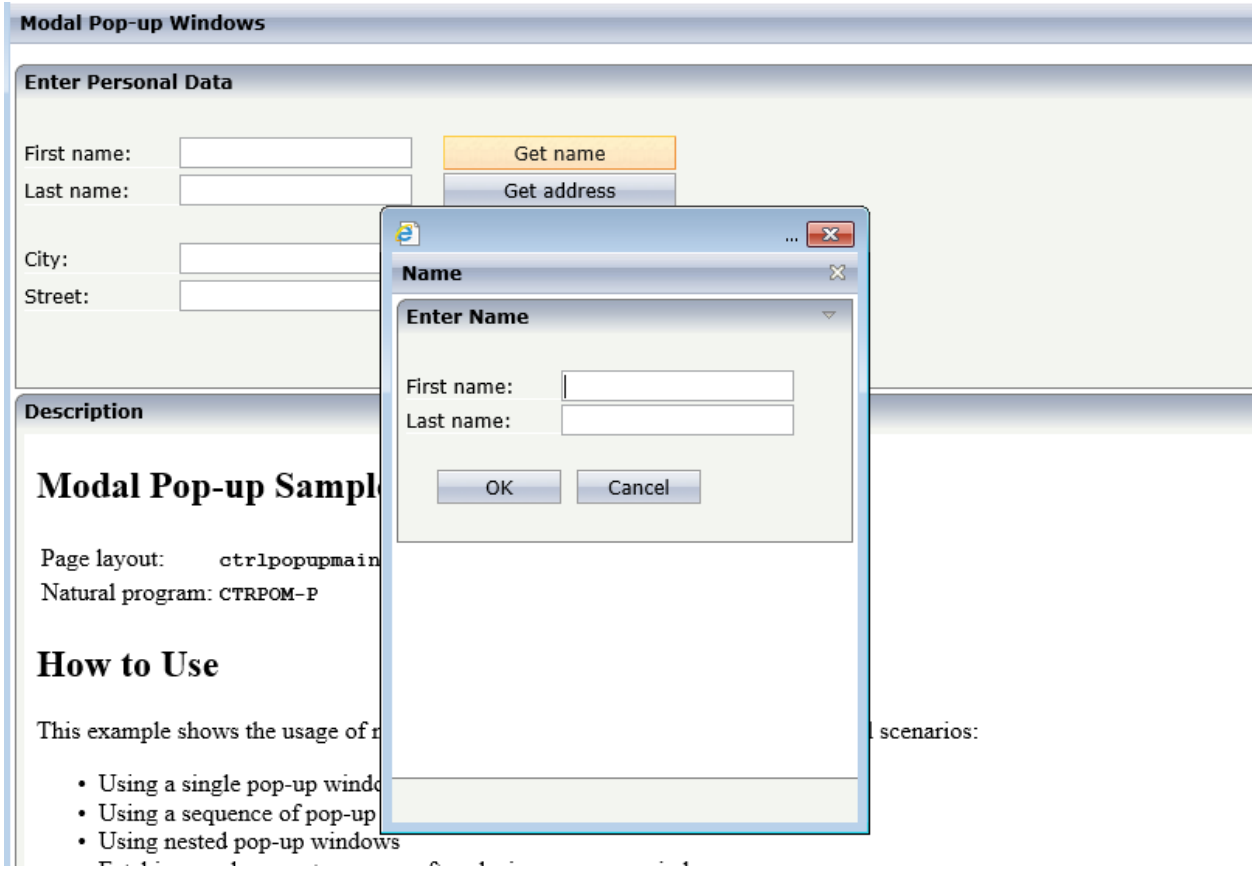


Special Considerations when using Internet Explorer 11

If you require your application to work with Internet Explorer 11 (IE11) only and you want to use the deprecated real modal pop-ups of IE11, you can activate this as described below. Please be aware, that the pop-ups then will not work in other browsers because most browsers have removed support for these kinds of pop-ups.

To activate the deprecated real modal pop-ups , add `"ie11only"` to the `popupfeatures` property in the layout of the dialog. For example, like this:

```
<natpage popupfeatures="dialogTop: SCRY(100)px; dialogLeft: SCRX(120)px;ie11only" ...
```



Pop-Ups - Responsive Pages

- Basics
- Internal Modals
- Dynamic Modals

Basics

You can open normal responsive pages as pop-ups. Use the PROCESS PAGE MODAL in your Natural program (see *Using Pop-Up Windows*). We call them *dynamic modals*.

In addition to supporting dynamic modals, responsive pages support a more lightweight form of modal called the *internal modal*. The layout definition is fully contained in the parent layout and the Natural data structures for the pop-up are also fully contained in the Natural program of the parent page (see *BMOBILE:INTMODAL*). Both modals - dynamic and internal - are always rendered as pure html within their parent browser window. No browser pop-ups are opened, therefore they are not a security risk.

Internal Modals

You can add multiple BMOBILE:INTMODAL controls to a page. Add a BMOBILE:INTMODAL-BODY control as child. In the BMOBILE:INTMODALBODY you can define your pop-up layout using the complete responsive control set in the usual way. To define how to open internal modals, use the properties of the BMOBILE:INTMODAL control.

Dynamic Modals

Each responsive page contains a control *BMOBILE:DYNMODAL* (= dynamic modal) which defines the rendering and behavior of pop-ups opened by this page. Therefore, all pages are capable of opening pop-ups and you can dynamically decide at runtime if your page uses pop-ups or not. You don't need to decide this at design time.

To define how to open dynamic modals, use any or all of the following:

1. Properties of the BMOBILE:DYNMODAL control in the parent page.
2. The properties `popupheight`, `popupwidth` and `popupfeature` in the ***NATPAGE*** control of the pop-up layout.
3. The control ***NJX:XCIOPENPOPUP*** in the parent layout.

If all three are used, 3 will overwrite the settings of 2 and 2 will overwrite the settings of 1.

IX Working with Workplaces

This part deals with applications that organize multiple pages in so-called workplaces. A prerequisite of building workplaces is an understanding of multi frame pages.

The information provided in this part is organized under the following headings:

What are Multi Frame Pages?

Definition of Multi Frame Pages

Application Designer Workplace Framework

Creating Your Own Workplace Application

Executing and Debugging Workplace Applications

Multi Language Management in Workplace Applications

NJX:XCIWPINFO2

NJX:XCIWPFUNCTIONS

NJX:XCIWPACCESS2

104 What are Multi Frame Pages?

Multi frame pages are a special set of pages. Normal pages represent a generated HTML page - a multi frame page represents a generated HTML frameset page.

A multi frame page does not contain controls but frames in which other pages are positioned. Each frame is associated with an ID (called “target” in this section). A frame may be:

- a normal HTML page
- an intelligent Application Designer page
- a frameset itself containing frames

Multi frame pages are the preferred way of arranging Application Designer pages in a frameset. Besides enhanced possibilities of communication between frames, multi frame pages automatically take care of keeping all Application Designer frames inside the same session.

105 Definition of Multi Frame Pages

- MFPAGE 778
- MFCISFRAME 779
- MFHTMLFRAME 782
- MFFRAMESET 783

The definition of multi frame pages is done with the Layout Painter. When you create a new layout, a dialog appears in which you select a template. To create a multi frame page, you have to select the "Multi Frame Page" template. The Layout Painter will open just as usual, but instead of having the PAGE control as the highest control, you now see the control MFPAGE. You can reach a number of controls that are related to multi frame page management.

The following controls are “normal frame controls” (they are described below):

- MFPAGE - the top element of multi frame pages.
- MFCISFRAME - a frame in which an Application Designer HTML page is loaded.
- MFHTMLFRAME - a frame in which a normal HTML page is loaded.
- MFFRAMESET - an area that can be subdivided into frames itself.

The following controls are “workplace controls” (they are described in the section *Application Designer Workplace Framework*. The Application Designer workplace is based on these controls.

- MFWPFUNCTIONS
- MFWPACTIVEFUNCTIONS
- MFWPCONTENT

MFPAGE

The MFPAGE is the top node of every multi frame page. It can be subdivided into frames or framesets.

Basic			
separation	Specifies how the corresponding internally used frameset is subdivided: choose "rows" for subdividing into rows, "cols" for subdividing into columns.	Obligatory	rows cols
sizing	Defines the size of the contained sub-frames. If you have three sub-frames to show up inside the page then you might specify "200,200,*" to specify how the height (if SEPARATION is "rows") or the width (if SEPARATION is "cols") is distributed among the frames. You can specify per frame either a pixel value or a "*".	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
border	Space between frames contained in the frameset that is internally built up.	Optional	1

			2 3 int-value
bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
frameborder	Defines if to display a border around the contained frames. Valid values are "true" or "false".	Optional	true false
framespacing	Defines the amount of additional space between the frames. Value is a pixel value.	Optional	1 2 3 int-value
framesetstyle	Style passed to the HTML-frameset definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

MFCISFRAME

The MFCISFRAME represents a frame in which an Application Designer page is shown. The name of the page is passed as a parameter.

Basic			
target	Id of the frame. Must be unique inside the frameset page. Must only contain alphanumeric characters. The id is important! CIS offers certain methods inside the Model-class that allow an adapter to start operations for a certain frame (e.g. openeCIPageInFrame(...)). As part of the parameters of these methods a target-id is passed. The target-id is exactly the id you specify with the TARGET property.	Obligatory	
cisurl	URL of the page to be shown inside. Use /project/page.html as syntax, e.g. "/HTMLBasedGUI/empty.html". Do NOT use only page.html believing that you do not have to specify the project because the multi frame page runs in the same project than the page you want to open - you ALWAYS have to specify the project!	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
resizable	Decision if the user is able to resize the frame. This property must be in synch with the definition in the "neighbour frames". If the neighbour frames do not support resizing then it will not be offered to the user as consequence. Valid values are "true" and "false". Default is "true".	Optional	true false
withborder	Boolean value defining if the frame has a border on its own. Default is "false".	Optional	true false
framestyle	Style that is passed to the HTML-FRAME definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080

			#000000
marginheight	Defines top and bottom margin height. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
marginwidth	Defines left and right margin width. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
withownborder	Flag that indicates if started pages show an own border. Default is false.	Optional	true false
Unload Behaviour			
unloadbehaviour	Reaction that CIS should take if the page inside the frame is closed. Possible values are "NOTHING" for doing nothing and "REMOVESESSION" for removing the session on server side. Do not define this property just "by accident" but leave it to the default ("NOTHING"). You only switch to "REMOVESESSION" if you want that the server side session is destroyed when leaving the page. This is the case if you have one page that clearly indicates the closing of a session at the point of time when the page is closed.	Optional	NOTHING REMOVESESSION

Applications can change the page that is shown inside the MFCISFRAME by using the method `Adapter.openCISPageInTarget(...)`.

MFHTMLFRAME

The MFHTMLFRAME represents a frame in which a normal HTML page is shown. This page can be a static HTML page or any URL - e.g. a URL referring to a certain JSP page.

Basic			
target	Id of the frame. Must be unique inside the frameset page. Must only contain alphanumeric characters. The id is important! CIS offers certain methods inside the Model-class that allow an adapter to start operations for a certain frame (e.g. openeCIPageInFrame(...)). As part of the parameters of these methods a target-id is passed. The target-id is exactly the id you specify with the TARGET property.	Obligatory	
url	URL to be opened inside the frame. The URL can be defined relative to the multi frame page or can be defined in an absolute way.. Example: You can define "../HTMLBasedGUI/workplace/header2.html" - or "http://www.softwareag.com".	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
resizable	Decision if the user is able to resize the frame. This property must be in synch with the definition in the "neighbour frames". If the neighbour frames do not support resizing then it will not be offered to the user as consequence. Valid values are "true" and "false". Default is "true".	Optional	true false
withborder	Boolean value defining if the frame has a border on its own. Default is "false".	Optional	true false
scrolling	Boolean that indicates whether the frame can be scrolled. Default is true.	Optional	true false
framestyle	Style that is passed to the HTML-FRAME definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
marginheight	Defines top and bottom margin height. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
marginwidth	Defines left and right margin width. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value

MFFRAMESET

The MFFRAMESET represents a frame that is internally again divided into frames. The MF-FRAMESET definition decides whether to divide into rows or columns, and how to size the inner frames.

Basic			
target	Id of the frame. Must be unique inside the frameset page. Must only contain alphanumeric characters. The id is important! CIS offers certain methods inside the Model-class that allow an adapter to start operations for a certain frame (e.g. openeCIPageInFrame(...)). As part of the parameters of these methods a target-id is passed. The target-id is exactly the id you specify with the TARGET property.	Obligatory	
separation	Specifies how the corresponding internally used frameset is subdivided: choose "rows" for subdividing into rows, "cols" for subdividing into columns.	Obligatory	rows cols

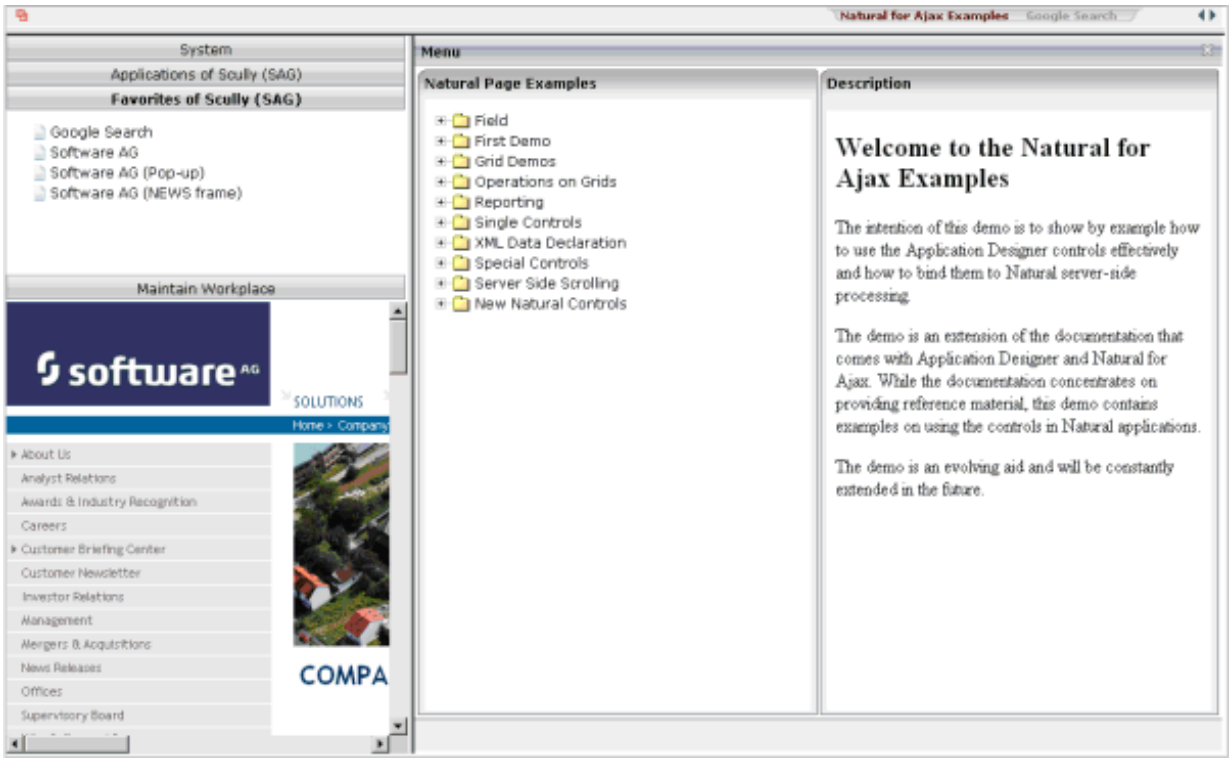
sizing	Defines the size of the contained sub-frames. If you have three sub-frames to show up inside the page then you might specify "200,200,*" to specify how the height (if SEPARATION is "rows") or the width (if SEPARATION is "cols") is distributed among the frames. You can specify per frame either a pixel value or a "*".	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
border	Space between frames contained in the frameset that is internally built up.	Optional	1 2 3 int-value
bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
frameborder	Defines if to display a border around the contained frames. Valid values are "true" or "false".	Optional	true false
framespacing	Defines the amount of additional space between the frames. Value is a pixel value.	Optional	1 2 3 int-value
framesetstyle	Style passed to the HTML-frameset definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

106 Application Designer Workplace Framework

- Framework Overview 786
- Functions Frame: MFWPFUNCTIONS 788
- Active Functions Frame: MFWPACTIVEFUNCTIONS 790
- Content Frame: MFWPCONTENT 791
- Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO 792
- Customizing the MFWPFUNCTIONS Behavior 803
- Session Management inside the Workplace 812
- Workplace API for Dynamic Manipulation 812

The Natural for Ajax demos provide an example of a workplace built on base of the Application Designer framework. The example can be executed with the following URL:

http://<host>:<port>/cisnatural/servlet/StartCISPage?PAGEURL=/njxdemos/wpdynworkplace.html

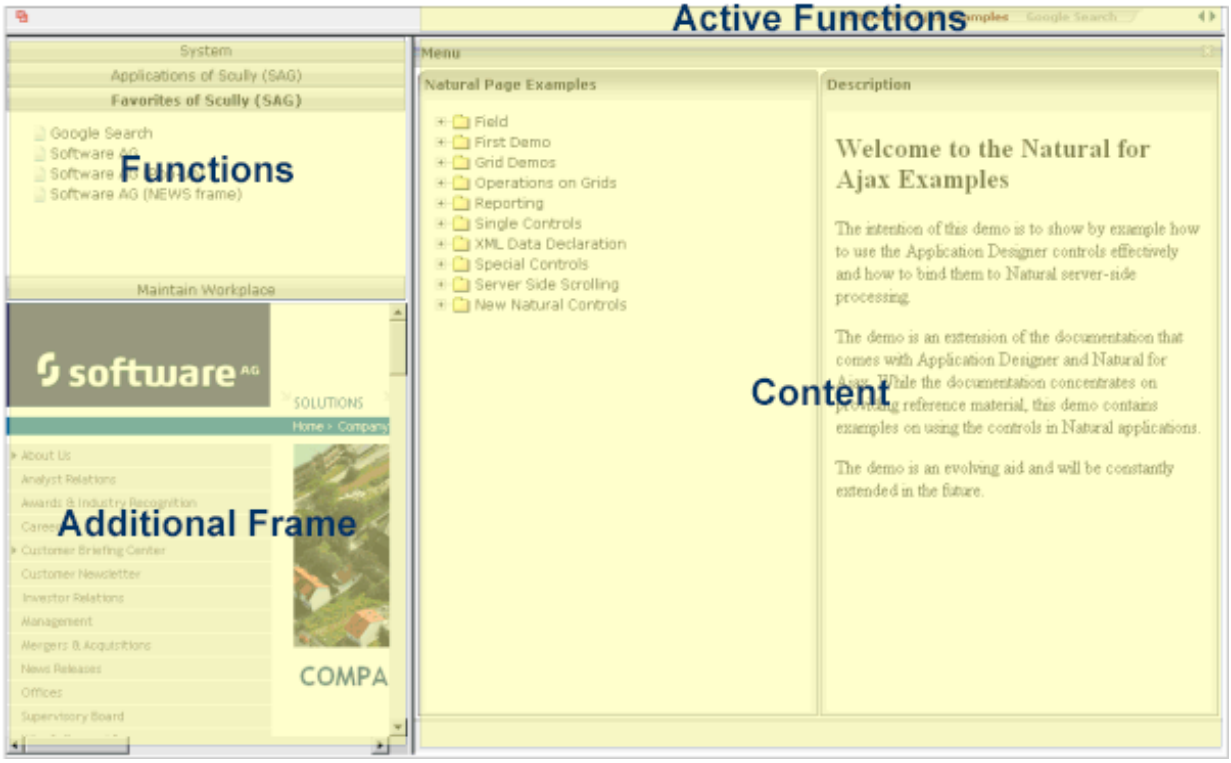


The workplace framework bases on the multi frame page management described in the previous sections. It offers the following:

- flexible arrangement of frames,
- dynamic loading of available functions,
- possibility to change the environment at runtime via specific controls,
- execution of multiple tasks between which the user can switch (“multi document interface”).

Framework Overview

An Application Designer workplace is a certain arrangement of frames in a multi frame page. Some of the frames have predefined tasks. Have a look at the example workplace in which you can already see the most important frames:



The "Functions" frame contains the available functions that can be chosen and invoked by the user. The "Content" frame contains the page or page sequence that is opened if a function is selected. The "Active Functions" frame shows the functions that were opened by the user and allows the user to navigate between the active functions.

Have a look at the XML layout definitions for this workplace; it defines how the frames are arranged (*../njx<nn>.ear/cisnatural.war/njxdemos/xml/wpdynworkplace.xml*):

```
<?xml version="1.0" encoding="UTF-8"?>
<mfpage separation="rows" sizing="20,*">
    <mfwpactivefunctions resizable="false" withborder="false" scrolling="false"
                         framestyle="border: 0px solid #000000">
    </mfwpactivefunctions>
    <mfframeset target="ZZZ" separation="cols" sizing="265,*">
       <mfframeset target="LEFTPART" separation="rows" sizing="*,400" border="true"
                   framesetstyle="border: 1px solid #808080">
           <mfwpfunctions bootstrapinfourl="/njxdemos/xml/wpdynbootstrapinfo.xml"
                         serversidescrolling="false" framestyle="border: 1 solid ↩
#808080;">
            </mfwpfunctions>
            <mfhtmlframe target="NEWS" url="../njxdemos/wpdynhowto.html"
                   resizable="true" withborder="false" scrolling="true"
                   framestyle="border: 1px solid #808080">
            </mfhtmlframe>
        </mfframeset>
        <mfwpcontent resizable="true" withborder="true" scrolling="false"
```

```
                    framestyle="border: 1 solid #808080;">
        </mfwpcontent>
    </mfframeset>
</mfpage>
```

You see that there are three special frame controls that are used internally: MFWPFUNCTIONS, MFWPACTIVEFUNCTIONS and MFWPCONTENT. In addition, there is one HTML page arranged below the MFWPFUNCTIONS control.

Let us take a closer look at each of the three workplace frame controls.

Functions Frame: MFWPFUNCTIONS

This is the frame to hold the available functions to be selected by the user. The control has the following properties:

Basic			
bootstrapclass	Name of the class that is responsible for passing the initial workplace configuration. The class must support interface "IMFWorkplace2" and must support a constructor without parameters. When being displayed the workplace creates an instance of this class and asks for an object that represents the workplace setup. Have a look into the javadoc-documentation for interface "IMFWorkplace2" for more information.	Optional	
bootstrapinfourl	URL to an .xml file that holds the initial workplace configuration. Do not use BOOTSTRAPINFOURL and BOOSTRAPCLASS at the same time! Use /project/directory/doc.xml as syntax, e.g. /HTMLBasedGUI/workplace/bootstrapworkplaceinfo.xml.	Optional	
serversidescrolling	Flag that decides if the function tree providing the available workplaces functions support client side scrolling (default, "false") or supports server side scrolling ("true"). Server side scrolling should be used if a function tree contains more than 100 nodes.	Optional	true false
defaultcontentpage	URL of a page that is shown in the 'content area' by default.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
contentstylesheet	Style sheet that should be used for the content that is started inside the workplace.	Optional	

framestyle	Style that is passed to the HTML-FRAME definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
marginheight	Defines top and bottom margin height. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
marginwidth	Defines left and right margin width. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
activefunctionsvariant	Defines how the MFWPACTIVEFUNCTIONS frame displays the list of started pages. You can either use a STRIPSEL or TABSTRIP control. Default is "tabstrip".	Optional	tabstrip stripsel
withownborder	Flag that indicates if the functions page shows an additional border. Default is false.	Optional	true false
workplacestylesheet	Style sheet that should be used for the workplace itself.	Optional	
withplusminus	If set to "true" then +/- Icons will be rendered in front of the mfwpfuntions.	Optional	true false
workplaceproject	If set to a valid project name, standard messages and standard dialogs used by the default workplace framework will be generated into this project. At runtime the messages and dialogs of this project will be used instead of the default	Optional	

	ones of the HTMLBasedGUI project. Generated multilanguage files:workplace and popups. Generated layouts:popupyesno and popupok.		

Active Functions Frame: MFWPACTIVEFUNCTIONS

This frame shows the functions that the user started and between which the user can switch.

Basic			
resizable	Decision if the user is able to resize the frame. This property must be in synch with the definition in the "neighbour frames". If the neighbour frames do not support resizing then it will not be offered to the user as consequence. Valid values are "true" and "false". Default is "true".	Optional	true false
withborder	Boolean value defining if the frame has a border on its own. Default is "false".	Optional	true false
scrolling	Boolean that indicates whether the frame can be scrolled. Default is true.	Optional	true false
framestyle	Style that is passed to the HTML-FRAME definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF #808080 #000000
marginheight	Defines top and bottom margin height. Value is a pixel value. Default is "0".	Optional	1 2 3

			int-value
marginwidth	Defines left and right margin width. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Content Frame: MFWPCONTENT

This is the frame in which content is started that is selected from the functions area.

Basic			
resizable	Decision if the user is able to resize the frame. This property must be in synch with the definition in the "neighbour frames". If the neighbour frames do not support resizing then it will not be offered to the user as consequence. Valid values are "true" and "false". Default is "true".	Optional	true false
withborder	Boolean value defining if the frame has a border on its own. Default is "false".	Optional	true false
scrolling	Boolean that indicates whether the frame can be scrolled. Default is true.	Optional	true false
framestyle	Style that is passed to the HTML-FRAME definition that is internally generated.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
bordercolor	Sets the border color of the frame set.	Optional	#FF0000 #00FF00 #0000FF #FFFFFF

			#808080 #000000
marginheight	Defines top and bottom margin height. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
marginwidth	Defines left and right margin width. Value is a pixel value. Default is "0".	Optional	1 2 3 int-value
withownborder	Flag that indicates if started pages show an own border. Default is false.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO

The MFWPFUNCTIONS frame can be filled initially by using the `bootstrapinfourl` property. This property expects an URL to an XML file that represents the initial workplace setup (for example, *../njx<nn>.ear/cisnatural.war/njxdemos/xml/wpdynworkplace.xml*).

Have a look at the corresponding XML file:

```
<mfwpbootstrapinfo
         defaultcontentpage="/HTMLBasedGUI/empty.html"
         workplacestylesheet="../cis/styles/CIS_DEFAULT.css"
         synchtabnavigation="true"
         showdustbin="true"
         withtakeouttopopup="false"
         withcloseallwindowsicon="false"
         ↩
mfworkplaceeventlistener="com.softwareag.cis.workplace.MFDefaultEventListener"
         targetnameofresizableleftpart="AVAILABLEACTIVITIES"
         translationproject="tshmfp"
         translationreference="mfworkplace">
```

```
  <mfwptopic
         name="System"
         treeclass="WORKPLACETOPIC1ClientTree">

    <mfwpfolder
         name="System"
         draginfo="System"
         opened="true">

      <mfwpopencispage
         name="Login"
        activityurl="/cisnatural/NatLogon.html&amp;xciParameters.natsession=Workplace
                      ↩
&amp;xciParameters.natparam=stack%3D%28LOGON+SYSEXNJX%3BWPLGIN-P%29"
         onlyoneinstance="true"
         followpageswitches="true">
      </mfwpopencispage>

    </mfwpfolder>

  </mfwptopic>

  <mfwptopic
       name="Maintain Workplace"
       treeclass="WORKPLACETOPIC1ClientTree">

    <mfwpopencispage
       name="Maintain Function Tree"
      activityurl="/cisnatural/NatLogon.html&amp;xciParameters.natsession=Workplace
                    ↩
&amp;xciParameters.natparam=stack%3D%28LOGON+SYSEXNJX%3BWPFUNC-P%29"
       onlyoneinstance="true"
       followpageswitches="true">
    </mfwpopencispage>

    <mfwpopencispage
       name="Maintain Content Pages"
      activityurl="/cisnatural/NatLogon.html&amp;xciParameters.natsession=Workplace
                    ↩
&amp;xciParameters.natparam=stack%3D%28LOGON+SYSEXNJX%3BWPCONT-P%29"
       onlyoneinstance="true"
       followpageswitches="true">
    </mfwpopencispage>

  </mfwptopic>

</mfwpbootstrapinfo>
```

Note: To make sure that you are using a proper *bootstrapinfo.xml* file, use the XML Schema *editor.xsd* (and all corresponding XSD files) to validate your XML file (for example, in XMLSpy).

Overview of the bootstrapinfo hierarchy:

```
<mfwpbootstrapinfo>          // root tag
    <mfwptopic>              // new topic
       <mfwpfolder>          // MFWorkplaceTreeNodeFolder
       <mfwpopencispage>     // MFWorkplaceTreeNodeCISPage
       <mfwpopencispopup>    // MFWorkplaceTreeNodeCISPopup
       <mfwpopencistarget>   // MFWorkplaceTreeNodeCISTarget
       <mfwpopenhtmlpage>    // MFWorkplaceTreeNodeHTMLPage
       <mfwpopenhtmlpopup>   // MFWorkplaceTreeNodeHTMLPopup
       <mfwpopenhtmltarget>  // MFWorkplaceTreeNodeHTMLTarget
```

`<mfwpfolder>` can contain each of the other `<mfwptopic>` subtags including itself.

The following topics are covered below:

- MFWPBOOTSTRAPINFO Properties
- MFWPTOPIC Properties
- MFWPFOLDER Properties
- MFWPOPENCISPAGE Properties
- MFWPOPENCISPOPUP Properties
- MFWPOPENCISTARGET Properties
- MFWPOPENHTMLPAGE Properties
- MFWPOPENHTMLPOPUP Properties
- MFWPOPENHTMLTARGET Properties

MFWPBOOTSTRAPINFO Properties

Basic			
defaultcontentpage	The workplace consists out of several frames, one of it the content frame. If there is no active activity in the workplace then the defaultContentPage is displayed inside the content frame. You can use this in two ways: (1) Either create one "background page" which always is shown in an "empty" workplace. (2) Or create one "background page" which the workplace opens by default. E.g. you want in a start-workplace to first present to the user a logon page. EXAMPLE: "/HTMLBasedGUI/empty.html"	Optional	

workplacestylesheet	The style sheet which is used for the left and top frame of the workplace. If no style sheet is specified then the workplace adapts to the standard style sheet which is kept in the CISsession context. You typically want to use one fix child for a workplace - because the workplace is typically embedded in some other frames arranging some graphics/etc. around, and you do not want the workplace colour's to change independent from this. EXAMPLE: "/cis/styles/XYZ_STYLE.css"	Optional	background-co #FF0000 color: #0000FF font-weight: bo
translationproject	Name of the project where the actual used multilanguage file is located. e.g. cisdemos	Optional	
translationreference	Name of the multilanguage .csv file. e.g. test (if the file test.csv should be used)	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
mfworkplaceeventlistener	Use this interface to react on workplace events. (1) Create an implementation of this interface (2) Use method `MFWorkplaceInfo.registerMFWorkplaceEventListener` to register your class (3) Use method `NODEInfo.setDropInfo` on each tree item to be able to drag that item Step two and three are typically done within the "bootstrap info provider"-class A CISworkplace is a certain arrangement of frames in a multi frame page. The "functions"-frame (MFWPFUNCTIONS) holds the available functions to be selected by the user (click with the left mouse Button). In addition you can provide for right mouse button menu or drag and drop within the function tree. With that you may allow users to add/remove/shift menu items (personalization).	Optional	
targetnameofresizableleftpart	The workplace may contain a favourite list. At the bottom of the favourite list there are some items by which you can influence the size of the corresponding left part of the workplace. The name of the target frame to be resized is passed with this method.	Optional	

View			
showdustbin	Flag that indicates whether the dustbin (have a look at the DEMO WORKPLACE) is shown or not. Boolean value, default is false.	Optional	true false
synchtabnavigation	Set flag that decides if the tree "on the left" is synchronized with the tab navigation "on the top". If the user selects an opened activity in the tab strip then the corresponding tree node and topic is shown as consequence. Pay attention: the base of the synchronization is the naming of nodes. There is currently no naming concept beyond (that e.g. assigns ids to nodes). Make sure, your tree nodes are set in a way that each one holds a unique name. Use the tabText (setTabText) in order to make nodes unique! true ==> synchronization is done; false ==> synchronization is not done; default is false.	Optional	true false
withcloseallwindowsicon	Flag that indicates whether the CloseAllWindowsIcon is shown in the workplace or not. Boolean value, default is false.	Optional	true false
withtakeouttopopup	Flag that indicates	Optional	true false
browsertitleappendix	Customize the browser title. An empty string means, that you don't want to set the activity title as browser title. A non-empty string means, that for each activity the browser title is set to a concatenated value of the activity title and the browsertitleappendix you specified. Example: My Activity - My Browser Title Appendix.	Optional	
helpframewidth	The width of the cis help frame in pixels. Use this property in case you are using the CISHELP frame in your workplace for showing online help pages.	Optional	100 120 140 160 180 200 50% 100%

MFWPTOPIC Properties

Basic			
name	Text of the topic.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
buttonstyle	Style info that is passed to the button representing the topic.	Optional	
iconurl	The button that represents this topic may have an additional icon in front of the text. Use this parameter to set the icon URL.	Optional	
treestyle	Background style for the tree. You can e.g. define background colors and background pictures. Avoid the usage of ' and " characters. Please also have a look onto the method "setStyleClass" - via this method you can pass a reference to a CSS class.	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
treeclass	Sets the style class for rendering the tree area of the topic. There are 10 standard style classes available in the default style sheet: PLACETOPIC1ClientTree to WORKPLACETOPIC10ClientTree. These style sheets can be maintained within the CISstyle sheet editor.	Optional	
tooltip	Tooltip of the node.	Optional	
tooltipid	Text ID of the tooltip.	Optional	

MFWPFOLDER Properties

Basic			
name	Text of the tree node folder.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	

opened	Flag that indicates whether the folder is opened or not. Boolean value	Optional	true false
tooltip	Text of the tooltip of the tree node folder.	Optional	
tooltipid	Text ID of the tooltip.	Optional	

MFWPOPENCISPAGE Properties

Basic			
name	Text of the node.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
activityurl	URL to be started when user clicks on node. You can append parameters to the URL by appending them via "&param1=value1&param2=value2"	Obligatory	
followpageswitches	If the user navigates inside the called page (e.g. switches from one page to the other) then this navigation is registered. True means: when reinvoking the page through the tree then the user come back exactly to the page where he/she stayed. False means: the user is brought back to the starting page always. For HTML pages: Registering of the navigation is only supported for HTML pages in the framebuffer. This means you need to set the framebuffersize attribute in the cisconfig.xml file correspondingly.	Obligatory	
onlyoneinstance	A page with the corresponding text is only started once inside the workplace. If the page already exists no new pages is started but the existing one is picked.	Obligatory	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	
iconurl	URL for the icon in front of the text. The workplace itself is running in project "HTMLBasedGUI" - you have to go up first "../" to address your icons.	Optional	
tooltip	Text of the tooltip of the tree node.	Optional	
tooltipid	Text ID of the tooltip.	Optional	

MFWPOPENCISPOPUP Properties

Basic			
name	Text of the node.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
activityurl	URL to be started when user clicks on node. You can append parameters to the URL by appending them via "&param1=value1&param2=value2"	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	
iconurl	URL for the icon in front of the text. Must start with "../project".	Optional	
tooltip	Tooltip of the node.	Optional	
tooltipid	Text ID of tooltip.	Optional	
width	Set the dimension of the popup in pixels. (width)	Optional	1 2 3 int-value
height	Set the dimension of the popup in pixels. (height)	Optional	1 2 3 int-value
left	Set the dimension of the popup in pixels. (left)	Optional	1 2 3 int-value
top	Set the dimension of the popup in pixels. (top)	Optional	1 2 3

			int-value

MFWPOPENCISTARGET Properties

Basic			
name	Text of the node.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
activityurl	URL to be started when user clicks on node. You can append parameters to the URL by appending them via "&param1=value1&param2=value2".	Obligatory	
target	Name of the target Frame in which the CIS page is going to be opened. During workplace definition each frame you define gets assigned a target-id.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	
iconurl	URL for the icon in front of the text. Must start with "../project".	Optional	
tooltip	Tooltip of the node.	Optional	
tooltipid	Text ID of the tooltip.	Optional	

MFWPOPENHTMLPAGE Properties

Basic			
name	Text of the node.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
activityurl	URL to be started when user clicks on node.	Optional	
followpageswitches	If the user navigates inside the called page (e.g. switches from one page to the other) then this navigation is registered. True means: when reinvoking the page through the tree then the user come back exactly to the page where he/she stayed. False means: the user is brought back to the starting page always. For HTML pages: Registering of the navigation is only supported for HTML pages in the framebuffer. This means you need to set the framebuffersize attribute in the cisconfig.xml file correspondingly.	Optional	

onlyoneinstance	A page with the corresponding text is only started once inside the workplace. If the page already exists no new pages is started but the existing one is picked.	Optional	true false
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	
iconurl	URL for the icon in front of the text. Must start with "../project"	Optional	
tooltip	Tooltip of the node.	Optional	
tooltipid	Text ID of the tooltip.	Optional	

MFWPOPENHTMLPOPUP Properties

Basic			
name	Text of the node.	Optional	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
activityurl	URL to be started when user clicks on node.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
iconurl	URL for the icon in front of the text. Must start with "../project"	Optional	
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	
tooltip	Tooltip of the node.	Optional	
tooltipid	Text ID of the tooltip.	Optional	
width	Set the dimension of the popup in pixels. (width)	Optional	1 2 3 int-value
height	Set the dimension of the popup in pixels. (height)	Optional	1 2 3 int-value

left	Set the dimension of the popup in pixels. (left)	Optional	1 2 3 int-value
top	Set the dimension of the popup in pixels. (top)	Optional	1 2 3 int-value

MFWPOPENHTMLTARGET Properties

Basic			
name	Text of the node.	Obligatory	
textid	Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime. Do not specify a "name" inside the control if specifying a "textid".	Optional	
activityurl	URL to be started when user clicks on node.	Obligatory	
target	Name of the target Frame in which the HTML Page is going to be opened. When defining a workplace page you assign a target-id per frame.	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
iconurl	URL for the icon in front of the text Must start with "../project".	Optional	
draginfo	Any information that is useful to react on a drop event. Characters ' and \ are not allowed.	Optional	
tooltip	Tooltip of the node.	Optional	
tooltipid	Text ID of the tooltip.	Optional	

Customizing the MFWPFUNCTIONS Behavior

The `mfworkplaceeventlistener` property of MFWPBOOTSTRAPINFO defines a Java class name. This class listens to events raised by the workplace and reacts accordingly. Examples for such events are context menu requests, or reactions to opening, closing, removing or switching of content pages. You can write your own event handler class by providing a Java class which implements the `com.softwareag.cis.workplace.IMFWorkplaceEventListener2` interface. See the Javadoc documentation (see also *Developing Java Extensions* in the *Ajax Developer* documentation).

Often, you do not want to write a complete event handler class. Instead, you would like to keep most of the default behavior, but simply customize pop-up messages and/or the shown context menus for the different nodes in the function tree. The following topics describe how to do simple customizations for the default event handler implementation.

You start with the class `com.softwareag.cis.workplace.MFCustomEventListener`. If you only want to customize pop-up messages, you can simply extend this class. If you would like to customize context menus and/or reactions to other events, you can use the `MFCustomEventListener` class as a template for writing your own custom event listener. The `MFCustomEventListener` class extends the `MFEventListenerBase` class which implements basic event reactions.

The following topics are covered below:

- Customizing Pop-up Messages
- Customizing Context Menus
- Implementing Custom Event Reactions (Advanced)
- Source Code for com.softwareag.cis.workplace.MFCustomEventListener

Customizing Pop-up Messages

If you only want to customize pop-up messages and keep the default context menu and event reaction, proceed as follows.

Create a class (for example, `MyCustomEventListener`) and implement the following methods (see also the example below):

- `String getPopupMessageNumberOfWorkplaceActivitiesReached(...)`
- `String getPopupTitelMaxNumberOfWorkplaceActivitiesReached(...)`
- `String getPopupMessagePopupMenuClosedByUser()`
- `String getPopupTitelPopupMenuClosedByUser()`

```
public class MyCustomEventListener extends MFCustomEventListener
{
protected String getPopupMessageNumberOfWorkplaceActivitiesReached(
                      int    maxactivities)
{
    return "THIS IS MY OWN MESSAGE";
}

protected String getPopupTitleNumberOfWorkplaceActivitiesReached(
                      int    maxactivities)
{
    return "THIS IS MY OWN POP-UP TITLE";
}

protected String getPopupMessagePopupMenuClosedByUser()
{
    return "THIS IS MY OWN MESSAGE";
}

protected String getPopupTitlePopupMenuClosedByUser()
{
    return "THIS IS MY OWN POP-UP TITLE";
}
}
```

Specify the `MyCustomEventListener` class in your bootstrapinfo (see below) and put the class file into the classpath of your web application.

```
<mfwpbootstrapinfo
    defaultcontentpage="/HTMLBasedGUI/empty.html"
         ...
         mfworkplaceeventlistener="com.mycompany.MyCustomEventListener"
...
```

Customizing Context Menus

If you would like to have your own context menus, you need to implement the following methods:

- `TREECollection buildContextMenu(...)`
- `TREECollection buildDropMenu(...)`
- `TREECollection buildFunctionContextMenu(...)`
- `TREECollection buildMFTopicContextMenu(...)`

All of these methods return a `TREECollection` object with the nodes for the context menu. For details of the different methods, see the corresponding Javadoc documentation of the `com.softwareag.cis.workplace.MFEventListenerBase` class. See also *Developing Java Extensions* in the *Ajax Developer* documentation.

Recommendation:

1. Write your own class (for example, `AnotherCustomEventListener`) which extends `MFEventListenerBase`.
2. Use the `MFCustomEventListener` class as a template. Here you can see how a `TREECollection` object is built. You can copy all required information and paste it in your own class.

A `TREECollection` is an object which describes a tree of nodes. Each node implements some standard commands such as **Remove**, **Cut** or **Paste**. If you look at the `MFCustomerEventListener` class, you will see the class `MFCustomMenuNodeInfo` which extends the class `MFMenuNodeInfoBase`. The `MFMenuNodeInfoBase` class contains the implementation of a set of standard commands which are defined as `CMDID_*` fields in the class. See the corresponding Javadoc documentation for details (see also *Developing Java Extensions* in the *Ajax Developer* documentation). You can reuse the standard commands, or you can implement your own commands.

Recommendation for implementing your own commands:

1. Write your own node class (for example, `MyCustomMenuNodeInfo`) which extends `MFMenuNodeInfoBase`.
2. In the same way as the `MFCustomEventListener` class builds the `TREECollection` objects from `MFCustomMenuNodeInfo` nodes, your `AnotherCustomEventListener` class will build the `TREECollection` objects from the `MyCustomMenuNodeInfo` nodes.

To use your newly implemented event listener class `AnotherCustomEventListener`, specify the `AnotherCustomEventListener` class in your bootstrapinfo (see below) and put the class file into the classpath of your web application.

```
<mfwpbootstrapinfo
    defaultcontentpage="/HTMLBasedGUI/empty.html"
         ...
      mfworkplaceeventlistener="com.mycompany.AnotherCustomEventListener"
...
```

Implementing Custom Event Reactions (Advanced)

If you also want to implement own reactions to other events, you create your own class (for example, `MyAdvancedEventListener`) which implements the interface `com.softwareag.cis.workplace.IMFWorkplaceEventListener2`. See the Javadoc documentation for details (see also *Developing Java Extensions* in the *Ajax Developer* documentation).

Your class must implement the `react*` methods of this interface:

```
public class MyAdvancedEventListener implements IMFWorkplaceEventListener2
{
   public void reactOnDrop(...){...}
   public Boolean reactOnCloseWindowRequest(...){...}
   ...
}
```

To add your `MyAdvancedEventListener` class to the bootstrapinfo, proceed in the same way as described in the previous topics.

Source Code for com.softwareag.cis.workplace.MFCustomEventListener

```
package com.softwareag.cis.workplace;

import com.softwareag.cis.server.util.TREECollection;

/**
 * This class is an example of a simple custom event listener based on the
 * <code>MFEventListenerBase</code> default implementation. The source code is
 * available in the documentation.
 * <p>
 * It shows how to simply customize pop-up messages, pop-up titles and/or context
 * menus without having to write a complete event listener.
 * <p>
 *
 * @see com.softwareag.cis.workplace.MFEventListenerBase
 *
 */
public class MFCustomEventListener extends MFEventListenerBase
{

    /**
     * Objects of this class represent a context menu item. It extends the
     * default implementation for context menu items {@link #MFMenuNodeInfoBase}.
     * This default implementation defines default items for the basic commands
     * like CUT, PASTE, REMOVE.
     * <p>
     *
     * @see com.softwareag.cis.workplace#MFMenuNodeInfoBase
     *
     */
    public class MFCustomMenuNodeInfo extends MFMenuNodeInfoBase
    {
        /**
         * Constructor
         *
         * @param eventListener the event listener
         */
        MFCustomMenuNodeInfo(MFEventListenerBase eventListener)
        {
            super(eventListener);
```

```
        }

        /* (non-Javadoc)
         * @see ↩
com.softwareag.cis.workplace.MFMenuNodeInfoBase#init(java.lang.String,
         *      java.lang.String, com.softwareag.cis.workplace.IMFWorkplace,
         *      com.softwareag.cis.server.util.TREECollection,
         *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral[],
         *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral,
         *      com.softwareag.cis.workplace.MFWorkplaceTopic)
         */
        protected void init(String text,
                            String image,
                            IMFWorkplace workplace,
                            TREECollection tree,
                            MFWorkplaceTreeNodeGeneral[] treeNodes,
                            MFWorkplaceTreeNodeGeneral treeNode2,
                            MFWorkplaceTopic topic)
        {
            super.init(text, image, workplace, tree, treeNodes, treeNode2, topic);
        }

        /* (non-Javadoc)
         * @see com.softwareag.cis.workplace.MFMenuNodeInfoBase#init(int,
         *      com.softwareag.cis.workplace.IMFWorkplace,
         *      com.softwareag.cis.server.util.TREECollection,
         *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral[],
         *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral,
         *      com.softwareag.cis.workplace.MFWorkplaceTopic)
         */
        protected void init(int cmdid,
                            IMFWorkplace workplace,
                            TREECollection tree,
                            MFWorkplaceTreeNodeGeneral[] treeNodes,
                            MFWorkplaceTreeNodeGeneral treeNode2,
                            MFWorkplaceTopic topic)
        {
            super.init(cmdid, workplace, tree, treeNodes, treeNode2, topic);
        }

    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#buildDropMenu(com.softwareag.cis.workplace.IMFWorkplace,
     *      com.softwareag.cis.workplace.MFWorkplaceTopic,
     *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral,
     *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral[])
     */
    protected TREECollection buildDropMenu(IMFWorkplace workplace,
```

```
                                          MFWorkplaceTopic topic,
                                          MFWorkplaceTreeNodeGeneral targetNode,
                                        MFWorkplaceTreeNodeGeneral[] droppedItems)
    {
        TREECollection menu = new TREECollection();
        MFCustomMenuNodeInfo menuNode = null;
        if (targetNode.getOpened() == 2)
        {
            menuNode = new MFCustomMenuNodeInfo(this);
            menuNode.init(MFMenuNodeInfoBase.CMDID_MOVEBEFORE, workplace, ↩
topic.getTree(), droppedItems, targetNode, topic);
            menu.addTopNode(menuNode, true);
            menuNode = new MFCustomMenuNodeInfo(this);
            menuNode.init(MFMenuNodeInfoBase.CMDID_MOVEBEHIND, workplace, ↩
topic.getTree(), droppedItems, targetNode, topic);
            menu.addTopNode(menuNode, true);
        }
        else
        {
            menuNode = new MFCustomMenuNodeInfo(this);
            menuNode.init(MFMenuNodeInfoBase.CMDID_MOVEASFIRST, workplace, ↩
topic.getTree(), droppedItems, targetNode, topic);
            menu.addTopNode(menuNode, true);
            menuNode = new MFCustomMenuNodeInfo(this);
            menuNode.init(MFMenuNodeInfoBase.CMDID_MOVEASLAST, workplace, ↩
topic.getTree(), droppedItems, targetNode, topic);
            menu.addTopNode(menuNode, true);
        }
        return menu;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#buildContextMenu(com.softwareag.cis.workplace.IMFWorkplace,
     *      com.softwareag.cis.workplace.MFWorkplaceTopic,
     *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral,
     *      com.softwareag.cis.workplace.MFWorkplaceTreeNodeGeneral[])
     */
    protected TREECollection buildContextMenu(IMFWorkplace workplace,
                                              MFWorkplaceTopic topic,
                                              MFWorkplaceTreeNodeGeneral item,
                                            MFWorkplaceTreeNodeGeneral[] selection)
    {
        TREECollection tree = topic.getTree();
        TREECollection menu = new TREECollection();

        // ---------------------- Show with sub menu
        MFCustomMenuNodeInfo menuNode = null;
        menuNode = new MFCustomMenuNodeInfo(this);
        menuNode.init(MFMenuNodeInfoBase.CMDID_SHOW, workplace, tree, selection, ↩
```

```
null, topic);
        menu.addTopNode(menuNode, false);

        MFCustomMenuNodeInfo subNode = null;
        subNode = new MFCustomMenuNodeInfo(this);
       subNode.init(MFMenuNodeInfoBase.CMDID_SHOW_CONTENT_FRAME, workplace, tree, ↩
selection, null, topic);
        menu.addSubNode(subNode, menuNode, true, false);

        subNode = new MFCustomMenuNodeInfo(this);
        subNode.init(MFMenuNodeInfoBase.CMDID_SHOW_NEW_WINDOW, workplace, tree, ↩
selection, null, topic);
        menu.addSubNode(subNode, menuNode, true, false);

        // --------------- CUT
        menuNode = new MFCustomMenuNodeInfo(this);
        menuNode.init(MFMenuNodeInfoBase.CMDID_CUT, workplace, tree, selection, ↩
null, topic);
        menu.addTopNode(menuNode, true);

        // --------------- PASTE
        menuNode = new MFCustomMenuNodeInfo(this);
       menuNode.init(MFMenuNodeInfoBase.CMDID_PASTE, workplace, tree, selection, ↩
null, topic);
        menu.addTopNode(menuNode, true);
        if (super.getClipboardSize() == 0 ||
            item.getOpened() == 2) menuNode.setInactive(true);

        // --------------- Separator
        menuNode = new MFCustomMenuNodeInfo(this);
       menuNode.init("&SEPARATOR", null, workplace, tree, selection, null, topic);
        menu.addTopNode(menuNode, true);

        // --------------- REMOVE
        menuNode = new MFCustomMenuNodeInfo(this);
       menuNode.init(MFMenuNodeInfoBase.CMDID_REMOVE, workplace, tree, selection, ↩
null, topic);
        menu.addTopNode(menuNode, true);

        return menu;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#buildFunctionContextMenu(com.softwareag.cis.workplace.IMFWorkplace,
     *      com.softwareag.cis.workplace.MFWorkplaceTopic)
     */
    protected TREECollection buildFunctionContextMenu(IMFWorkplace workplace,
                                                      MFWorkplaceTopic selectedTopic)
    {
```

```
        TREECollection menu = new TREECollection();
        MFCustomMenuNodeInfo menuNode = null;

        menuNode = new MFCustomMenuNodeInfo(this);
        menuNode.init(MFMenuNodeInfoBase.CMDID_REFRESHTOPIC, workplace, ↩
selectedTopic.getTree(), null, null, selectedTopic);
        menu.addTopNode(menuNode, true);

        menuNode = new MFCustomMenuNodeInfo(this);
        menuNode.init(MFMenuNodeInfoBase.CMDID_REMOVEALL, workplace, ↩
selectedTopic.getTree(), null, null, selectedTopic);
        menu.addTopNode(menuNode, true);

        return menu;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#buildMFTopicContextMenu(com.softwareag.cis.workplace.IMFWorkplace,
     *      com.softwareag.cis.workplace.MFWorkplaceTopic)
     */
    protected TREECollection buildMFTopicContextMenu(IMFWorkplace workplace,
                                                     MFWorkplaceTopic selectedTopic)
    {
        TREECollection menu = new TREECollection();
        MFCustomMenuNodeInfo menuNode = null;

        menuNode = new MFCustomMenuNodeInfo(this);
        menuNode.init(MFMenuNodeInfoBase.CMDID_REFRESHTOPIC, workplace, ↩
selectedTopic.getTree(), null, null, selectedTopic);
        menu.addTopNode(menuNode, true);
        return menu;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#getMaxNumberActivitiesMode()
     */
    protected int getMaxNumberActivitiesMode()
    {
        return MAX_NUMBER_ACTIVITIES_POPUP;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#getPopupMessageNumberOfWorkplaceActivitiesReached(int)
```

```
     */
    protected String getPopupMessageNumberOfWorkplaceActivitiesReached(int ↩
maxactivities)
    {
        // use default
        return null;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#getPopupTitelMaxNumberOfWorkplaceActivitiesReached(int)
     */
    protected String getPopupTitelMaxNumberOfWorkplaceActivitiesReached(int ↩
maxactivities)
    {
        // use default
        return null;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#getPopupMessagePopupMenuClosedByUser()
     */
    protected String getPopupMessagePopupMenuClosedByUser()
    {
        // use default
        return null;
    }

    /*
     * (non-Javadoc)
     *
     * @see ↩
com.softwareag.cis.workplace.MFEventListenerBase#getPopupTitelPopupMenuClosedByUser()
     */
    protected String getPopupTitelPopupMenuClosedByUser()
    {
        // use default
        return null;
    }

}
```

Session Management inside the Workplace

When the user selects functions in the MFWPFUNCTIONS frame, then pages are opened in the content frame, or as pop-ups or in a named target frame.

The workplace offers a “multi document interface” - i.e. you can work in parallel in several activities and you can switch between these activities. This structure is reflected in the server-side session structure. The section *Details on Session Management* in the *Special Development Topics* (which is part of the Application Designer documentation) explains this in a detailed way. However, some information is given below.

The session management of Application Designer knows sessions (typically representing a browser instance) and subsessions (reflecting a user activity with a defined life cycle). A session contains one or more subsessions. Inside one subsession, the adapter object are kept which are required by a page or a page sequence. Subsessions are isolated from one another.

The workplace proceeds in the following way:

- Every activity that is started inside the content is represented by a subsession of its own. If you have opened five Application Designer pages via the function tree inside the content area of the workplace, then there are five subsessions on the server side. If the user navigates between the activities (e.g. via the MFWPACTIVEFUNCTIONS frame), then from session point of view the user navigated between subsessions.
- The workplace itself also occupies one subsession. If Application Designer pages are opened in a pop-up or in a named target, then these pages are living inside the subsession of the workplace.

When programming content pages, you do not notice the session management: every page that you design and test in the Layout Painter behaves in the same way in the workplace. Due to the separation into subsessions, you are not aware of "neighboring" subsessions.

Workplace API for Dynamic Manipulation

Internally, the workplace is started when the workplace frameset page is loaded. So far you got to know the framework to set up the workplace by providing a MFWPBOOTSTRAPINFO file.

But you can also dynamically manipulate the workplace. There are two typical usages:

- You can exchange all workplace definitions dynamically. Maybe you offer the user a “reduced” workplace just allowing the user to log on at the beginning. Afterwards, the “real” workplace for the user is built up - containing all functions available for the user.

- You can manipulate workplace definitions in an existing workplace. For example, you modify the title of an activity that is shown in the MFWPACTIVEFUNCTIONS area. Or you want to add certain nodes to an existing tree.

For this purpose, there is a set of controls containing the workplace functions that you can use from your application:

- **NJX:XCIWPINFO2**
- **NJX:XCIWPFUNCTIONS**
- **NJX:XCIWPACCESS2**

107 Creating Your Own Workplace Application

- Elements of a Workplace Application 816
- Using the Default Workplace Framework 816
- Customizing the Frames, Dialogs and Messages of the Default Workplace Framework 817
- Using Your Own Standard Pop-up Dialogs and Messages 817
- Using Your Own Active Functions Frame 819

Elements of a Workplace Application

When you create your own workplace application, it consists of several different elements:

- The workplace framework
  - Standard workplace frames and pop-up dialogs
  - Standard multi-language messages and texts
  - Standard behavior
- A specific application
  - Page layouts with Natural/Java implementation
  - Multi-language messages and texts
  - Style
  - Behavior

Depending on the application needs, it might be convenient to use the default implementation of the workplace framework as it is. Other applications might want to adapt the look-and-feel of the default workplace frames, standard pop-up dialogs and/or texts to their own needs. The topics below describe both approaches.

Using the Default Workplace Framework

The default workplace framework is the workplace implementation as used in the development workplace itself. Standard dialogs and frames used in this implementation are not part of your project. They reside in the central project **HTMLBasedGUI** and are simply used from within your project.

To create a workplace application with the default look-and-feel, proceed as follows:

1. Use the Layout Painter to create a new layout which uses the "Multi Frame Workplace" layout template (this template is located on the **Workplace** tab).

   Customize all required properties to your needs. However, leave the `workplaceproject` property of the **MFWPFUNCTIONS** control empty.
2. Create an MFWPBOOTSTRAPINFO XML file. See *Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO*.

   As the value for the `mfworkplaceeventlistener` property, specify "com.softwareag.cis.workplace.MFCustomEventListener". This is a ready-to-use default implementation. You can also add your own event listener, see *Customizing the MFWPFUNCTIONS Behavior*.

Customizing the Frames, Dialogs and Messages of the Default Workplace Framework

As shown in the section *Framework Overview*, the workplace framework contains the following predefined frames which can be customized slightly via the corresponding properties:

- Functions
- Active Functions
- Contents

In addition, the workplace framework uses own messages and standard dialogs from the central **HTMLBasedGUI** project. The files of the **HTMLBasedGUI** project are not intended to be modified because any modifications will be lost during the next product upgrade. Instead, you can use your own "Active Functions" frame and/or your own standard pop-up dialogs and standard messages as described below.

Using Your Own Standard Pop-up Dialogs and Messages

Open your "Multi Frame Workplace" page layout in the Layout Painter. Set the value of the property `workplaceproject` to the name of your project. When saving the page layout, the following files will be generated into your project:

- `<myproject>`*/multilanguage/de/workplace.csv*
- `<myproject>`*/multilanguage/en/workplace.csv*
- `<myproject>`*/multilanguage/de/popups.csv*
- `<myproject>`*/multilanguage/en/popups.csv*
- `<myproject>`*/popupok.xml*
- `<myproject>`*/popupyesno.xml*

The *workplace.csv* files in the *multilanguage* directory contain standard messages used in the workplace framework for English and German. Be careful not to change the text IDs. However, you can adapt the texts and you can also add additional languages.

The names of the *popups.csv* files in the *multilanguage* directory are defined by the `translationreference` property. See the sample layout below.

The *popupok.xml* and *popupyesno.xml* files are the layouts for the used standard pop-up dialogs. Be sure not to change the names of the layouts, the corresponding adapter classes or the proper-

ties/methods. You can adapt the layouts but you must keep the following parts that are shown in bold below:

```
<page ispopup="true"
      model="com.softwareag.cis.popups.PopupOKModel"
      popupheight="170" popupwidth="310"
      translationreference="popups">
  <pagebody takefullheight="true">
    <itr height="100%" width="100%">
      <textout align="center" valign="middle"
               valueprop="question" width="100%">
      </textout>
    </itr>
    <vdist height="5">
    </vdist>
    <itr align="center">
      <button method="reactOnOK" textid="popupok.button0"
              withtd="false">
      </button>
    </itr>
    <vdist height="8">
    </vdist>
  </pagebody>
</page>
```

```
<page model="com.softwareag.cis.popups.PopupYesNoModel"
      translationreference="popups" popupwidth="310"
      popupheight="200">
  <pagebody takefullheight="true">
    <vdist height="5">
    </vdist>
    <itr width="100%" align="center">
    <textout valueprop="question" width="100%" height="100"
             align="center">
    </textout>
    </itr>
    <vdist height="5">
    </vdist>
    <itr align="center">
      <button textid="btnYes" method="reactOnYes">
      </button>
      <hdist width="10">
      </hdist>
      <button textid="btnNo" method="reactOnNo">
      </button>
    </itr>
    <vdist height="5">
    </vdist>
  </pagebody>
</page>
```

Using Your Own Active Functions Frame

To create and use your own "Active Functions" frame, proceed as follows:

1. Use the Layout Painter to create a new layout which uses one of the following layout templates (these templates are located on the **Workplace** tab):

   Workplace Activities Frame
   Workplace Activities Frame 2
   Workplace Stripsel Activities Frame
   Workplace Stripsel Activities Frame 2

   The layout templates either use a **TABSTRIP2** or a **STRIPSEL** control to render the activities. Both variants are available with and without navigation frame icons ("Close All Windows", "Reopen Navigation Frame", "Close Navigation Frame"). The layout templates which contain the number 2 in their names include the additional navigation frame icons.
2. You can modify the generated layout, but you must keep the parts that are shown in bold below.

```
<page
  model="com.softwareag.cis.workplace.MFActivitiesAdapter"
  translationreference="workplace"
  ...
  <tabstrip2 tabstripprop="activities2"
  ...
```

   For controlling the navigation frame, you can use the following method names:

   - `onOSize` (close the navigation frame)
   - `onNormalSize` (reopen the navigation frame)
   - `onCloseAll` (close all windows)

   Note that the generated layout is implemented by the Java adapter `com.softwareag.cis.workplace.MFActivitiesAdapter`. This Java adapter will handle these events. The "Active Function" frame cannot be implemented by a NATPAGE control. Also note that the events listed above are not passed to the Natural application.
3. In the **MFWPFUNCTIONS** control of your "Multi Frame Workplace" layout, set the `activefunctionsvariant` property to the URL of your own "Active Functions" frame layout. Example:

```
activefunctionsvariant="/myproject/myactiveFunctionsFrame.html"
```

108 Executing and Debugging Workplace Applications

- Prerequisites 822
- Creating a Session Configuration on the Application Server 822
- Configuring the Session in Your Workplace Application 823
- Executing the Workplace Application in NaturalONE 823
- Debugging the Workplace Application in NaturalONE 823
- Using Private-mode Libraries in Workplaces 825

Prerequisites

If you want to execute and debug workplace applications with NaturalONE, you have to install a Natural Web I/O Interface server (NWO). The Natural Web I/O Interface server must run in the same environment (`FNAT`, `FUSER`, `FDIC` and `FSEC`) as the Natural Development Server (NDV) which is used in NaturalONE.

For detailed information, see the section *Installing and Configuring the Natural Web I/O Interface Server* in the *Natural Web I/O Interface* documentation, which is part of the Natural documentation for the different platforms.

Creating a Session Configuration on the Application Server

This section describes how to create a session using NaturalONE's internal Tomcat server.

If you are using the internal Tomcat server of NaturalONE, make sure that identical session definitions for the workplace are defined in both: in the Tomcat server of your NaturalONE installation and in the Natural for Ajax web server installation outside of NaturalONE.

To create a session

1 Start NaturalONE.

2 Find out on which port the internal Tomcat server of your NaturalONE installation is running.

When the console output for the internal Tomcat server is enabled in the Natural preferences, you can see the port in the **Console** view when starting NaturalONE. Look for a message such as the following:

```
Starting internal Tomcat server using HTTP port 8080 ...
```

The port varies if several instances of NaturalONE are running in parallel. Normally, it is 8080.

3 Use the following URL to invoke the configuration tool of Natural for Ajax on this port:

```
http://localhost:8080/cisnatural/conf_index.jsp
```

4 Invoke the **Session Configuration** page and create the required session.

For detailed information, see *Client Configuration* in the documentation for the runtime version of Natural for Ajax. The latest version of the Natural for Ajax documentation (which is not part of the NaturalONE documentation) can always be found at ***https://documentation.softwareag.com/***.

Configuring the Session in Your Workplace Application

After the session has been defined in the configuration tool, you have to configure this session in the XML file which represents your initial workplace setup. For detailed information, see ***Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO***.

Executing the Workplace Application in NaturalONE

After you have installed and configured the Natural Web I/O Interface server and after the session has been defined in the configuration tool, you can execute your workplace application using NaturalONE's Ajax Developer.

To execute the workplace application

1. In the **Project Explorer** view, select the XML layout which represents your workplace application.
2. Invoke the context menu and choose **Ajax Developer > Execute**.

Debugging the Workplace Application in NaturalONE

After you have installed and configured the Natural Web I/O Interface server and after the session has been defined in the configuration tool, you can debug your workplace application. For debugging, you can start the workplace application in two different ways: you can start it from within NaturalONE using the context menu, or you can start it directly in the browser using URL parameters. This is described in the topics below.

The prerequisite for debugging, however, is that the debug attach server has been enabled in the Natural preferences and that at least one breakpoint has been set in a Natural program which belongs to your workplace application. For further information, see *Using a Debug Attach Server* in *Using NaturalONE*.

The following topics are covered below:

- Starting the Workplace Application from NaturalONE

- Starting the Workplace Application Directly in the Browser

Starting the Workplace Application from NaturalONE

The following steps assume that you have already created a Natural project containing the relevant sources for the workplace application in NaturalONE.

To debug a workplace application

1 Open the source editor for a Natural program which belongs to the workplace application and set at least one breakpoint.

2 In the **Project Explorer** view, go to the **xml** folder which contains your page layouts.

3 Select the page layout for your workplace application (a page of type MFPAGE).

4 Invoke the context menu and choose **Ajax Developer > Debug**.

See also *Executing and Debugging User Interface Components* in the *Ajax Developer* documentation.

The workplace application is now started in the browser, in debug mode. When the program in which the breakpoint has been set is about to be executed inside the corresponding Natural session, the debugger is launched and the application stops at the first breakpoint. You can now use the **Debug** perspective as described in the section *Debugging Natural Applications* in *Using Natural ONE*.

Starting the Workplace Application Directly in the Browser

The parameters which are normally defined in the Natural preferences (host name and port number) and in the properties of a Natural project (client ID) can also be specified as URL parameters of the workplace application URL.

The following URL parameters are available for debugging purposes:

URL Parameter	Description
`xciParameters.natdebugclientid`	The client ID which has been generated for the project or, if defined, a custom client ID. You can find this ID in the project properties. See also *Debug Attach Settings* in *Using NaturalONE > Setting the Preferences*.
`xciParameters.natdebughost`	The name of the host (or IP address) on which the debug attach server is running. Do not use "localhost" as the host name.
`xciParameters.natdebugport`	The number of the port to which the debug attach server is listening.

Example:

http://localhost:8080/cisnatural/servlet/StartCISPage?PAGEURL=/njxdemos/wpdynworkplace.html&xciParameters.natdebugclientid=DC66D909-6872-4AFB-8898-D12B286BBE27&xciParameters.natdebughost=natqts43.eur.ad.sag&xciParameters.natdebugport=9999

Using Private-mode Libraries in Workplaces

Executing Workplace applications in private mode with Natural Security is fully supported. Without Natural Security, the support is limited.

- With Natural Security
- Without Natural Security

With Natural Security

Before executing or debugging the Workplace application, perform the following steps:

- Generate the LSO (library search order) container library

  If any private-mode library has been created for the project, the Natural node of the project properties offers a button to generate the LSO container library. After creating the LSO container library, its name displays. Private-mode batch execution requires this name. For further information, see *Managing Natural Projects* > *Changing the Project Properties* in *Using NaturalONE*.

- Save the private-mode library search order

  Select from the project's context menu:

  NaturalONE > Save Private-mode Library Search Order

  This action saves the current private-mode library search order into the LSO member. An LSO container library generates automatically should it not yet exist. The library's name can be found in the project properties, as described above.

Notes

1. Whenever the private-mode library search order changes, save the private-mode library search order again. For example, the private-mode library search order changes when adding a new private-mode library or when changing the steplibs.
2. NaturalONE offers a function to delete an LSO container library. For further information, see *Managing Natural Projects* > *Changing the Project Properties* in *Using NaturalONE*.

When executing or debugging the Workplace application, the name of the LSO container library will be added automatically as a URL parameter to the Workplace application. The Natural for Ajax runtime will automatically use this information for all activity URLs of the Workplace.

Example:

LSO container library name:

```
ABC00001
```

STACK statement of a workplace activity URL:

```
stack=(LOGON+MYLIB;MYPROG;FIN)
```

Dynamically adapted STACK statement by the Natural for Ajax runtime:

```
stack=(LOGON+MYLIB;SYSLSO+ABC00001;MYPROG;FIN)
```

Without Natural Security

Do NOT generate an LSO (library search order) container library. In case one already exists in your project, delete it. For further information, see *Managing Natural Projects* > *Changing the Project Properties* in *Using NaturalONE*.

When a project contains private mode libraries but no LSO container library, the private mode names are automatically used when executing or debugging the Workplace. Corresponding private mode names are automatically added as URL parameters. The Natural for Ajax runtime will automatically use this information for all activity URLs of the Workplace. Do not use additional STEBLIB definitions defined for the project or in Natural profiles of the execution environment.

Example:

Private mode name for library MYLIB:

```
XYZ00001
```

STACK statement of a workplace activity URL:

```
stack=(LOGON+MYLIB;MYPROG;FIN)
```

Dynamically adapted STACK statement by the Natural for Ajax runtime:

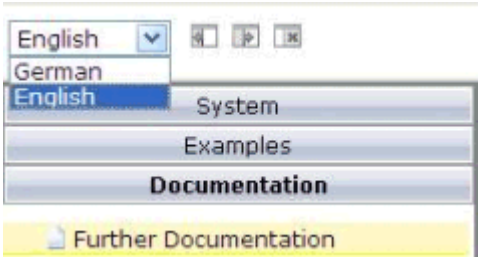
```
stack=(LOGON+XYZ00001;MYPROG;FIN)
```

109 Multi Language Management in Workplace Applications

- General Information 828
- Language Switch in Content Pages 829
- Language Switch in Function Tree and Activities Pane 829

General Information

The following provides some hints for multi language management in workplace applications which contain Natural layout pages. We assume that the user interface contains a combo box or another control which allows the end user to select the language in a Java-based layout page.



This small Java-based layout page can be very simple:

```
<?xml version="1.0" encoding="UTF-8"?>
<page model="com.softwareag.cis.test.workplace.LanguageSelectionAdapter"
    occupiedpixelheight="0" occupiedpixelwidth="0" ↩
contextmenumethod="processAsDefault">
    <pagebody vscroll="hidden" hscroll="hidden" horizdist="false">
        <vdist height="8">
        </vdist>
        <itr takefullwidth="true">
            <coltable0 width="100">
                <itr>
                    <hdist width="5">
                    </hdist>
                    <combofix valueprop="language" width="90" flush="server"
                        flushmethod="onLanguageChanged">
                        <combooption name="German" value="de">
                        </combooption>
                        <combooption name="English" value="en">
                        </combooption>
                    </combofix>
                </itr>
            </coltable0>
        </itr>
        <itr>
        </itr>
    </pagebody>
</page>
```

For an example of how to integrate such a page layout into a frameset of a workplace definition, see */njxdemos/xml/wpworkplacelan1.xml*. For language selection, this file uses the layout */njx-demos/xml/wplanselect1.xml*. Based on the user selection, the workplace application has to respond correspondingly. Depending on the application needs, the reactions can be different. The following

sections provide information on some basic reactions. Your workplace application may implement its own specific handling.

Language Switch in Content Pages

When the user selects a different language, the workplace application basically has two choices:

- either close the already started activities after asking or warning the user, or
- force the already started activities to switch to the newly selected language.

The `MFWorkplaceSessionUtil` class in the package `com.softwareag.cis.workplace` contains methods for performing both kinds of reaction.

Regardless of the variant you implement for the currently started activities, you want new layout pages to be started with the newly selected language. If your layout pages are implemented with Natural, use the configuration tool that is delivered with the Natural for Ajax product and set the **Language** option to **Set in workplace**. This makes sure that newly added or opened activities use the newly selected language.

Language Switch in Function Tree and Activities Pane

Sometimes, the newly selected language should only be applied to the activities in the workplace. Sometimes, the workplace application also wants to change the language in the function tree and activity pane.

If you use a static workplace definition (that is, a *bootstrapinfo.xml* file) and want to change the language in the function tree later, you must use `textid` properties instead of names.

You can change the language for a static workplace definition by reloading the workplace definition. The `MFWorkplaceSessionUtil` class in the package `com.softwareag.cis.workplace` contains convenient methods for reloading the workplace definition. When reloading a function tree, make sure to

- either close the started activities first, or
- update the page titles in the activities pane.

If you modify the function tree dynamically using the **NJX:XCIWPINFO2** and/or **NJX:XCIWP-FUNCTIONS** controls and want to change the language in the function tree later, you must also use `textid` properties in these controls.

To update the page titles in already started activities, you simply call the method `replaceLiteralInPageTitles()` of the `MFWorkplaceSessionUtil` class.

To change the language in a dynamically defined function tree, you simply call the method `replaceLiteralInFunctionTree()` in the `MFWorkplaceSessionUtil` class. Instead of reloading the workplace definition, you can also use this method for statically defined function trees.

See also the **wpworkplacelan2** example in the Natural for Ajax demos.

110 NJX:XCIWPINFO2

- Example 832
- Adapter Interface 832

The NJX:XCIWPINFO2 control is used to access and exchange the function tree that is shown in the "Functions" frame (**MFWPFUNCTIONS**) as a whole. In order to perform incremental changes in the function tree, you should use the **NJX:XCIWPFUNCTIONS** control.

The NJX:XCIWPINFO2 control provides a functional API to the workplace. It does not have design time properties nor does it raise events.

Example

The XML code for the example looks as follows:

```
<natpage xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <njx:xciwpinfo2>
    </njx:xciwpinfo2>
</natpage>
```

Adapter Interface

```
1 XCIWPINFO_CHANGEINDEX (I4)
1 XCIWPINFO_NODE (1:*)
2 ACTIVITYID (U) DYNAMIC
2 ACTIVITYURL (U) DYNAMIC
2 BUTTONSTYLE (U) DYNAMIC
2 DRAGINFO (U) DYNAMIC
2 FOLLOWPAGESWITCHES (L)
2 HEIGHT (I4)
2 ICONURL (U) DYNAMIC
2 LEFT (I4)
2 LEVEL (I4)
2 NAME (U) DYNAMIC
2 ONLYONEINSTANCE (L)
2 OPENED (I4)
2 TARGET (U) DYNAMIC
2 TEXTID (U) DYNAMIC
2 TOOLTIP (U) DYNAMIC
2 TOOLTIPID (U) DYNAMIC
2 TOP (I4)
2 TREECLASS (U) DYNAMIC
2 TREESTYLE (U) DYNAMIC
2 TYPE (U) DYNAMIC2 WIDTH (I4)
```

Each occurrence in the array `XCIWPINFO_NODE` describes a node in the function tree. The function tree consists of up to three levels: topics, folders and nodes.

Topic

The following structure elements are used to describe a topic:

Element	Meaning
BUTTONSTYLE	Style info that is passed to the button representing the topic.
ICONURL	The button that represents this topic may have an additional icon in front of the text. Use this parameter to set the icon URL.
LEVEL	The following definition means “This is a topic”: LEVEL = 1
NAME	Name of the topic.
TEXTID	Multi language dependent text that is displayed inside the control. The TEXTID is translated into a corresponding string at runtime.
OPENED	The following definition means “The topic is closed”: OPENED = 0 The following definition means “The topic is opened”: OPENED = 1
TOOLTIP	Text of the tooltip for the topic.
TOOLTIPID	Multi language dependent text that is displayed inside the control. The TOOLTIPID is translated into a corresponding string at runtime.
TREECLASS	Set the style class for rendering the tree area of the topic. There are ten standard style classes available in the default style sheet: PLACETOPIC1ClientTree to WORKPLACETOPIC10ClientTree. These style sheets can be maintained with the style sheet editor of the Application Designer.
TREESTYLE	Background style for the tree. For example, you can define background colors and background pictures. Avoid the usage of single quote (') and double-quote (") characters.

Folder

The following structure elements are used to describe a folder:

Element	Meaning
DRAGINFO	Any information that is useful to react on a drop event. The single quote (') and backslash (\) characters are not allowed.
LEVEL	The following definitions mean “This is a folder”: LEVEL >= 2 and OPENED = 0 or LEVEL >= 2 and OPENED = 1
NAME	Name of the folder.

Element	Meaning
TEXTID	Multi language dependent text that is displayed inside the control. The TEXTID is translated into a corresponding string at runtime.
OPENED	The following definition means "The folder is closed": OPENED = 0 The following definition means "The folder is opened": OPENED = 1
TOOLTIP	Text of the tooltip for the folder.
TOOLTIPID	Multi language dependent text that is displayed inside the control. The TOOLTIPID is translated into a corresponding string at runtime.

Node that opens a page in the "Content" frame

The following structure elements are used to describe a node that opens an Application Designer page or HTML page in the "Content" frame:

Element	Meaning
ACTIVITYURL	The URL to be loaded when the user clicks on a node. You can append parameters to the URL.
DRAGINFO	Any information that is useful to react on a drop event. The single quote (') and backslash (\) characters are not allowed.
FOLLOWPAGESWITCHES	If true, the workplace keeps the information when the user switches inside the content area from one page to the next. If the user reinvokes the page, the page to which the user switched last is shown, not the one from the ACTIVITYURL. The use of FOLLOWPAGESWITCHES only makes sense if ONLYONEINSTANCE is set to true. The following applies for HTML pages: Registering of the navigation is only supported for HTML pages in the frame buffer. This means that you have to set the framebuffersize parameter in the *cisconfig.xml* file correspondingly.
ICONURL	The URL for the icon which is shown in front of the name.
LEVEL	The following definition creates a node on level 2, that is, directly under a topic: LEVEL = 2 and OPENED = 2 The following definition creates a node on level 3, that is, under a folder: LEVEL = 3 and OPENED = 2
NAME	Name of the node.
TEXTID	Multi language dependent text that is displayed inside the control. The TEXTID is translated into a corresponding string at runtime.
ONLYONEINSTANCE	A page with the corresponding name is only started once inside the workplace. If the page already exists, no new page is started but the existing one is used.

Element	Meaning
OPENED	See the above description for LEVEL.
TOOLTIP	Text of the tooltip for the tree node.
TOOLTIPID	Multi language dependent text that is displayed inside the control. The TOOLTIPID is translated into a corresponding string at runtime.
TYPE	"cis" to open an Application Designer page, or "html" to open an HTML page.

Node that opens a page in a pop-up window

The following structure elements are used to describe a node that opens an Application Designer page or HTML page in a pop-up window:

Element	Meaning
ACTIVITYURL	The URL to be loaded when the user clicks on a node. You can append parameters to the URL.
DRAGINFO	Any information that is useful to react on a drop event. The single quote (') and backslash (\) characters are not allowed.
HEIGHT	Set the dimension of the pop-up in pixels.
ICONURL	The URL for the icon which is shown in front of the name.
LEFT	Set the relative position of the pop-up in pixels.
LEVEL	The following definition creates a node on level 2, that is, directly under a topic: LEVEL = 2 and OPENED = 2 The following definition creates a node on level 3, that is, under a folder: LEVEL = 3 and OPENED = 2
NAME	Name of the node.
TEXTID	Multi language dependent text that is displayed inside the control. The TEXTID is translated into a corresponding string at runtime.
OPENED	See the above description for LEVEL.
TOOLTIP	Text of the tooltip for the tree node.
TOOLTIPID	Multi language dependent text that is displayed inside the control. The TOOLTIPID is translated into a corresponding string at runtime.
TOP	Set the relative position of the pop-up in pixels.
TYPE	"cispopup" to open an Application Designer page, or "htmlpopup" to open an HTML page.
WIDTH	Set the dimension of the pop-up in pixels.

Node that opens a page in a target frame

The following structure elements are used to describe a node that opens an Application Designer page or HTML page in a target frame other than the “Content” frame:

Element	Meaning
ACTIVITYURL	The URL to be loaded when the user clicks on a node. You can append parameters to the URL.
DRAGINFO	Any information that is useful to react on a drop event. The single quote (') and backslash (\) characters are not allowed.
ICONURL	The URL for the icon which is shown in front of the name.
LEVEL	The following definition creates a node on level 2, that is, directly under a topic: LEVEL = 2 and OPENED = 2 The following definition creates a node on level 3, that is, under a folder: LEVEL = 3 and OPENED = 2
NAME	Name of the node.
TEXTID	Multi language dependent text that is displayed inside the control. The TEXTID is translated into a corresponding string at runtime.
OPENED	See the above description for LEVEL.
TARGET	Name of the target frame in which the page is to be opened. During workplace definition, you assign a target ID to each frame you define.
TOOLTIP	Text of the tooltip for the tree node.
TOOLTIPID	Multi language dependent text that is displayed inside the control. The TOOLTIPID is translated into a corresponding string at runtime.
TYPE	"cistarget": Open an Application Designer page. "htmltarget": Open an HTML page.

When the structure is passed to the application, it contains the information about the current function tree. The application may change this information and return it. In order to indicate that the function tree shall be updated in the user interface, the application must modify the value of XCIWPINFO_CHANGEINDEX on return. This is achieved, for instance, by the following statement:

```
ADD 1 TO XCIWPINFO_CHANGEINDEX
```

111 NJX:XCIWPFUNCTIONS

- Example 838
- Adapter Interface 838

The NJX:XCIWPFUNCTIONS control is used to modify the function tree that is shown in the "Functions" frame (**MFWPFUNCTIONS**) incrementally. In order to access the content of the function tree or to exchange it as a whole, you have to use the **NJX:XCIWPINFO2** control.

The NJX:XCIWPFUNCTIONS control provides a functional API to the workplace. It does not have design time properties nor does it raise events.

Example

The XML code for the example looks as follows:

```
<natpage xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <njx:xciwpfunctions>
    </njx:xciwpfunctions>
</natpage>
```

Adapter Interface

```
1 XCIWPFUNCTIONS (1:*)
2 CMDADDFOLDER
3 ADDFOLDER_ASFIRST (L)
3 ADDFOLDER_FOLDERNAME (U) DYNAMIC
3 ADDFOLDER_FOLDERTEXTID (U) DYNAMIC
3 ADDFOLDER_OPENED (I4)
3 ADDFOLDER_TOPICNAME (U) DYNAMIC
3 ADDFOLDER_TOPICTEXTID (U) DYNAMIC
2 CMDADDNODE
3 ADDNODE_ACTIVITYID (U) DYNAMIC
3 ADDNODE_ACTIVITYURL (U) DYNAMIC
3 ADDNODE_ASFIRST (L)
3 ADDNODE_FOLDERNAME (U) DYNAMIC
3 ADDNODE_FOLDERTEXTID (U) DYNAMIC
3 ADDNODE_HEIGHT (I4)
3 ADDNODE_LEFT (I4)
3 ADDNODE_NAME (U) DYNAMIC
3 ADDNODE_TARGET (U) DYNAMIC
3 ADDNODE_TEXTID (U) DYNAMIC
3 ADDNODE_TOP (I4)
3 ADDNODE_TOPICNAME (U) DYNAMIC
3 ADDNODE_TOPICTEXTID (U) DYNAMIC
3 ADDNODE_TYPE (U) DYNAMIC
3 ADDNODE_WIDTH (I4)
2 CMDADDTOPIC
3 ADDTOPIC_SWITCHTOTOPIC (L)
3 ADDTOPIC_TOPICNAME (U) DYNAMIC
```

```
3 ADDTOPIC_TOPICTEXTID (U) DYNAMIC
3 ADDTOPIC_TREECLASS (U) DYNAMIC
2 CMDREMFOLDER
3 REMFOLDER_FOLDERNAME (U) DYNAMIC
3 REMFOLDER_FOLDERTEXTID (U) DYNAMIC
3 REMFOLDER_TOPICNAME (U) DYNAMIC
3 REMFOLDER_TOPICTEXTID (U) DYNAMIC
2 CMDREMNODE
3 REMNODE_FOLDERNAME (U) DYNAMIC
3 REMNODE_FOLDERTEXTID (U) DYNAMIC
3 REMNODE_NAME (U) DYNAMIC
3 REMNODE_TEXTID (U) DYNAMIC
3 REMNODE_TOPICNAME (U) DYNAMIC
3 REMNODE_TOPICTEXTID (U) DYNAMIC
2 CMDREMTOPIC
3 REMTOPIC_TOPICNAME (U) DYNAMIC
3 REMTOPIC_TOPICTEXTID (U) DYNAMIC
```

Each occurrence in the array `XCIWPFUNCTIONS` describes a command that is to be sent to the workplace API. Several commands can be sent in a sequence. For each command, a corresponding substructure must be filled.

Add a topic

The following structure elements belong to `CMDADDTOPIC`:

Element	Meaning
`ADDTOPIC_SWITCHTOTOPIC`	"true": Open the new topic.
`ADDTOPIC_TOPICNAME`	Name of the topic.
`ADDTOPIC_TOPICTEXTID`	Multi language dependent text that is displayed inside the control. The `ADDTOPIC_TOPICTEXTID` is translated into a corresponding string at runtime.
`ADDTOPIC_TREECLASS`	Sets the style class for rendering the tree area of the topic. There are ten standard style classes available in the default style sheet: `PLACETOPIC1ClientTree` to `WORKPLACETOPIC10ClientTree`. These style sheets can be maintained with the style sheet editor of the Application Designer.

Add a folder

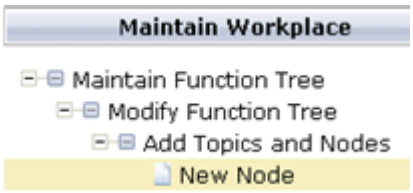
The following structure elements belong to `CMDADDFOLDER`:

Element	Meaning
ADDFOLDER_ASFIRST	"true": Add this folder as the first folder under the given topic. "false": Add this folder as the last folder under the given topic.
ADDFOLDER_FOLDERNAME	Name of the folder.
ADDFOLDER_FOLDERTEXTID	Multi language dependent text that is displayed inside the control. The ADDFOLDER_FOLDERTEXTID is translated into a corresponding string at runtime.
ADDFOLDER_OPENED	0: Add the folder as a closed folder with potential subnodes. 1: Add the folder as an opened folder. 2: Add the folder as a closed folder without subnodes.
ADDFOLDER_TOPICNAME	Name of the topic to which the folder is to be added.
ADDFOLDER_TOPICTEXTID	Multi language dependent text that is displayed inside the control. The ADDFOLDER_TOPICTEXTID is translated into a corresponding string at runtime.

Add a node

You can add all kinds of topic subnodes with the CMDADDNODE structure. Since a function tree can have any depth, you can precisely define the parent node for which you would like to add the new node as a child node. This can be done by specifying corresponding XPATH-like path information in the ADDNODE_FOLDERNAME or ADDNODE_FOLDERTEXTID element.

Example:



To add the node named "New Node", specify the following:

```
ADDNODE_TOPICNAME="Maintain Workplace"
ADDNODE_FOLDERNAME = "Maintain Function Tree/Modify Function Tree/Add Topics ↩
and Nodes"
ADDNODE_NAME="New Node"
```

The following structure elements belong to CMDADDNODE:

Element	Meaning
ADDNODE_NAME	Name of the node to be added.
ADDNODE_TEXTID	Multi language dependent text that is displayed inside the control. The ADDNODE_TEXTID is translated into a corresponding string at runtime.
ADDNODE_FOLDERNAME	Name of the folder to which the node is to be added.
ADDNODE_FOLDERTEXTID	Multi language dependent text that is displayed inside the control. The ADDNODE_FOLDERTEXTID is translated into a corresponding string at runtime.
ADDNODE_TOPICNAME	Name of the topic that contains this folder.
ADDNODE_TOPICTEXTID	Multi language dependent text that is displayed inside the control. The ADDNODE_TOPICTEXTID is translated into a corresponding string at runtime.
ADDNODE_ASFIRST	"true": Add this node as the first node under the given folder. "false": Add this node as the last node under the given folder.
ADDNODE_ACTIVITYURL	The URL to be loaded when the user clicks on the node. You can append parameters to the URL.
ADDNODE_ACTIVITYID	Use this element if you want to start different pages with the same name.
ADDNODE_TYPE	"cis": A node that opens an Application Designer page in the “Content” frame. "html": A node that opens an HTML page in the “Content” frame. "cispopup": A node that opens an Application Designer page in a pop-up window. "htmlpopup": A node that opens an HTML page in a pop-up window. "cistarget": A node that opens an Application Designer page in a target frame other than the “Content” frame. "htmlpopup": A node that opens an HTML page in a target frame other than the “Content” frame.
ADDNODE_LEFT	Only with type "cispopup" and "htmlpopup". Set the relative position of the pop-up in pixels.
ADDNODE_TOP	Only with type "cispopup" and "htmlpopup". Set the relative position of the pop-up in pixels.
ADDNODE_HEIGHT	Only with type "cispopup" and "htmlpopup". Set the dimension of the pop-up in pixels.
ADDNODE_WIDTH	Only with type "cispopup" and "htmlpopup". Set the dimension of the pop-up in pixels.
ADDNODE_TARGET	Only with type "cistarget" and "htmltarget". Name of the target frame in which the page is to be opened. During workplace definition, you assign a target ID to each frame you define.

Remove a topic

The following structure elements belong to CMDREMTOPIC:

Element	Meaning
REMTOPIC_TOPICNAME	Name of the topic to be removed.
REMTOPIC_TOPICTEXTID	Multi language dependent text that is displayed inside the control. The REMTOPIC_TOPICTEXTID is translated into a corresponding string at runtime.

Remove a folder

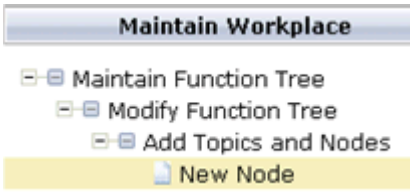
The following structure elements belong to CMDREMFOLDER:

Element	Meaning
REMFOLDER_FOLDERNAME	Name of the folder to be removed.
REMFOLDER_FOLDERTEXTID	Multi language dependent text that is displayed inside the control. The REMFOLDER_FOLDERTEXTID is translated into a corresponding string at runtime.
REMFOLDER_TOPICNAME	Name of the topic that contains the folder.
REMFOLDER_TOPICTEXTID	Multi language dependent text that is displayed inside the control. The REMFOLDER_TOPICTEXTID is translated into a corresponding string at runtime.

Remove a node

You can remove all kinds of topic subnodes with the CMDREMNODE structure. Since a function tree can have any depth, you can precisely define which node to remove by specifying corresponding XPATH-like path information in the REMNODE_FOLDERNAME or REMNODE_FOLDERTEXTID element.

Example:



To remove the node named "New Node", specify the following:

```
REMNODE_TOPICNAME="Maintain Workplace"
REMNODE_FOLDERNAME = "Maintain Function Tree/Modify Function Tree/Add Topics ↩
and Nodes"
REMNODE_NAME="New Node"
```

The following structure elements belong to CMDREMNODE:

Element	Meaning
REMNODE_FOLDERNAME	Name of the folder that contains the node to be removed.
REMNODE_FOLDERTEXTID	Multi language dependent text that is displayed inside the control. The REMNODE_FOLDERTEXTID is translated into a corresponding string at runtime.
REMNODE_NAME	Name of the node to be removed.
REMNODE_TEXTID	Multi language dependent text that is displayed inside the control. The REMNODE_TEXTID is translated into a corresponding string at runtime.
REMNODE_TOPICNAME	Name of the topic that contains the folder with the node to be removed.
REMNODE_TOPICTEXTID	Multi language dependent text that is displayed inside the control. The REMNODE_TOPICTEXTID is translated into a corresponding string at runtime.

112 NJX:XCIWPACCESS2

- Example 846
- Adapter Interface 846

The NJX:XCIWPACCESS2 control is used to open, activate and close content pages in the workplace, to open pages as pop-up windows, or to open pages in a frame.

This control provides a functional API to the workplace. It does not have design time properties nor does it raise events.

Example

The XML code for the example looks as follows:

```
<natpage xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
    <njx:xciwpaccess2>
    </njx:xciwpaccess2>
</natpage>
```

Adapter Interface

```
1 XCIWPACCESS2 (1:*)
  2 CMDADDPAGETOWORKPLACE
    3 ADD_ACTIVITYID (U) DYNAMIC
    3 ADD_ACTIVITYURL (U) DYNAMIC
    3 ADD_NAME (U) DYNAMIC
    3 ADD_TEXTID (U) DYNAMIC
    3 ADD_TYPE (U) DYNAMIC
  2 CMDCLOSECONTENTPAGE (U) DYNAMIC
  2 CMDINVOKEMETHODINCONTENTPAGE
    3 METHOD (U) DYNAMIC
  2 CMDOPENPAGEINTARGET
    3 OPEN_ACTIVITYURL (U) DYNAMIC
    3 OPEN_TARGET (U) DYNAMIC
    3 OPEN_TYPE (U) DYNAMIC
  2 CMDOPENPOPUP
    3 POPUP_ACTIVITYURL (U) DYNAMIC
    3 POPUP_HEIGHT (I4)
    3 POPUP_LEFT (I4)
    3 POPUP_TITLE (U) DYNAMIC
    3 POPUP_TITLEID (U) DYNAMIC
    3 POPUP_TOP (I4)
    3 POPUP_TYPE (U) DYNAMIC
    3 POPUP_WIDTH (I4)
  2 CMDSHOWPAGEINWORKPLACE
    3 SHOW_ACTIVITYID (U) DYNAMIC
    3 SHOW_ACTIVITYURL (U) DYNAMIC
    3 SHOW_NAME (U) DYNAMIC
```

```
    3 SHOW_TEXTID (U) DYNAMIC
    3 SHOW_TYPE (U) DYNAMIC
```

Each occurrence in the array XCIWPACCESS2 describes a command that is to be sent to the workplace API. Several commands can be sent in a sequence. For each command, a corresponding substructure must be filled.

Open a page in the "Content" frame

The following structure elements belong to CMDADDPAGETOWORKPLACE:

Element	Meaning
ADD_ACTIVITYURL	The URL to be loaded.
ADD_ACTIVITYID	Use this element if you want to start different pages with the same name.
ADD_NAME	The name to be displayed in the "Active Functions" frame.
ADD_TEXTID	Multi language dependent text that is displayed inside the control. The ADD_TEXTID is translated into a corresponding string at runtime.
ADD_TYPE	"cis": Open an Application Designer page. "html": Open an HTML page.

Open a page in a pop-up window

The following structure elements belong to CMDOPENPOPUP:

Element	Meaning
POPUP_ACTIVITYURL	The URL to be loaded. You can append parameters to the URL.
POPUP_TITLE	Title of the pop-up window.
POPUP_TITLEID	Multi language dependent text that is displayed inside the control. The POPUP_TITLEID is translated into a corresponding string at runtime.
POPUP_TYPE	"cis": Open an Application Designer page. "html": Open an HTML page.
POPUP_LEFT	Set the relative position of the pop-up in pixels.
POPUP_TOP	Set the relative position of the pop-up in pixels.
POPUP_WIDTH	Set the dimension of the pop-up in pixels.
POPUP_HEIGHT	Set the dimension of the pop-up in pixels.

Open a page in a target frame other than the "Content" frame

The following structure elements belong to CMDOPENPAGEINTARGET:

Element	Meaning
OPEN_ACTIVITYURL	The URL to be loaded. You can append parameters to the URL.
OPEN_TARGET	Name of the target frame in which the page is to be opened. During workplace definition, you assign a target ID to each frame you define.
OPEN_TYPE	"cis": Open an Application Designer page. "html": Open an HTML page.

Activate an already open page in the "Content" frame

The following structure elements belong to CMDSHOWPAGEINWORKPLACE:

Element	Meaning
SHOW_ACTIVITYURL	The URL to be loaded. You can append parameters to the URL.
SHOW_ACTIVITYID	Use this element if you want to start different pages with the same name.
SHOW_NAME	Name of the page in the “Active Functions” frame.
SHOW_TEXTID	Multi language dependent text that is displayed inside the control. The SHOW_TEXTID is translated into a corresponding string at runtime.
SHOW_TYPE	"cis": Activate an Application Designer page. "html": Activate an HTML page.

Close the currently active page in the "Content" frame

Assign the value "closeit" to CMDCLOSECONTENTPAGE.

Close all pages in the "Content" frame

Assign the value "all" to CMDCLOSECONTENTPAGE.

Or assign the value "allpopup" to CMDCLOSECONTENTPAGE. In this case, a yes/no pop-up will appear, asking whether you really want to close all content pages.

Invoke a method (raise an event) in the currently active page in the "Content" frame

The following structure element belongs to CMDINVOKEMETHODINCONTENTPAGE:

Element	Meaning
METHOD	Name of the method/event.

X Working with PDF Documents

113 Working with PDF Documents

- General Information 852
- About the Adapter Listener 852
- Example 853
- Built-in Events 854
- Advanced Data Binding and Rendering 855

General Information

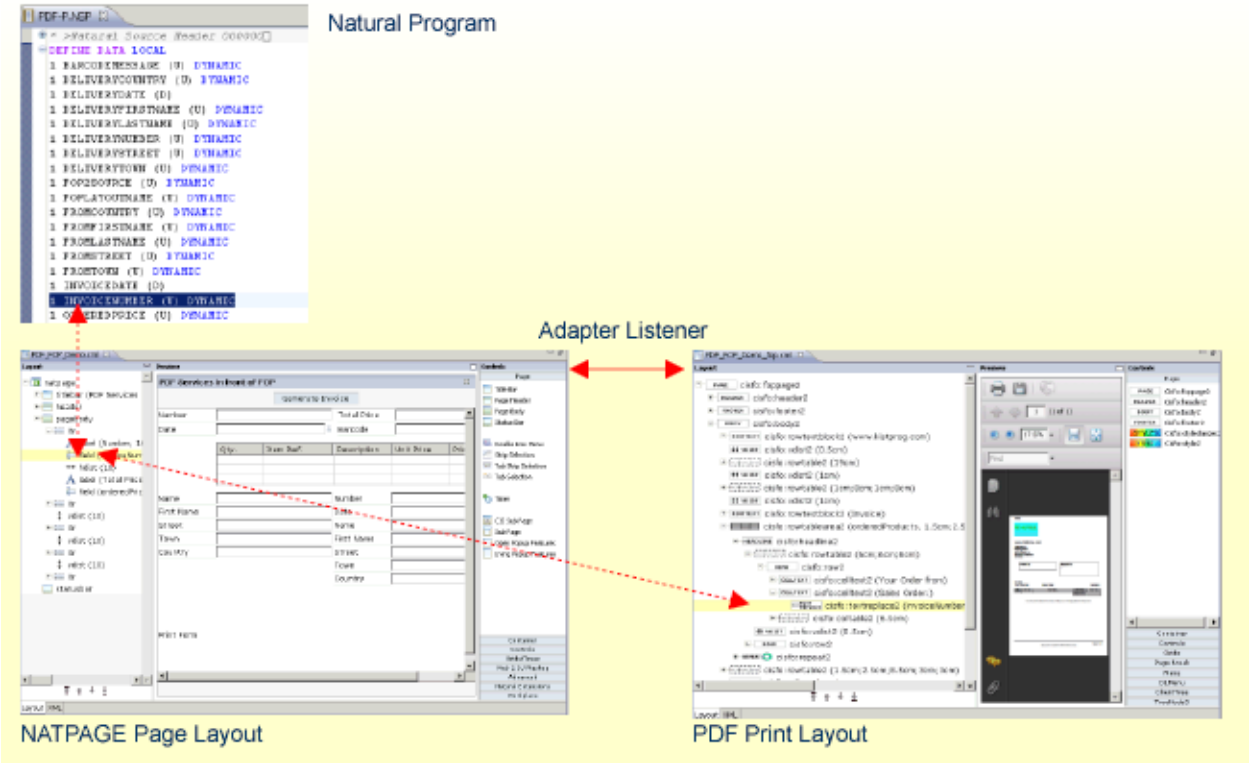
There are 2 different ways to generate PDF documents for Natural page layouts:

- **REPORT and REPORT2 control**
  You can fill a report dynamically at runtime from within your Natural program. According to the defined report, a PDF document is generated. This approach defines the look and feel of the PDF at runtime and therefore requires some programming in the corresponding Natural programs. For further information, see the description of the **REPORT** and the **REPORT2** control.

  The REPORT2 control can be used with responsive and non responsive pages. In difference to the REPORT control, the REPORT2 control does not require to additionally render the data in the page itself. It can be used to simply trigger the generation of a PDF report from a Java or Natural program.
- **Adapter Listener (`com.softwareag.cis.server.adapterlistener.PDFFOPListener`)**
  You can define the look and feel of the PDF document at design time. To do this, you create a specific PDF print form using the Layout Painter. This approach defines the PDF in a descriptive way and requires no programming on the Natural side. This approach is described in the topics below.

About the Adapter Listener

Using the adapter listener `com.softwareag.cis.server.adapterlistener.PDFFOPListener`, you can bind a **NATPAGE** page layout to a PDF print layout. The data in the PDF layout is descriptively bound to the properties of the NATPAGE page layout at design time. The NATPAGE page layout and the PDF print layout share the same data which is provided from the Natural program.



Example

The following steps describe how to add PDF support to a NATPAGE. You will find a running example in the Natural for Ajax demos.

1. In your NATPAGE page layout, set the property `adapterlisteners` to "com.softwareag.cis.server.adapterlistener.PDFFOPListener". If you have already applied another adapter listener, append the new listener using a semicolon.

```
<natpage ↩
adapterlisteners="com.softwareag.cis.server.adapterlistener.PDFFOPListener"
...
```

2. Add a button or an icon (or any other control that binds to a method) to the NATPAGE page layout, and as method apply the name "onPrintPDF".

```
<button name="Generate Invoice" method="onPrintPDF">
</button>
...
```

3. Create a PDF print form in the same project as your NATPAGE page layout. The name of the print form must be the same as the name of your page, appended with "_fop". For example, if the name of your page is "PDF_FOP_Demo.xml", the name of the PDF print form must be "PDF_FOP_Demo_fop.xml".
4. In the PDF print form, you use specific CISFO controls to define the rendering and the binding to the data. Use the Layout Painter to add and modify the corresponding CISFO controls. For detailed information on these controls, see *PDF Page Layout*.

   For the *PDF_FOP_Demo_fop.xml* layout, no Natural adapter will be generated. The data binding between the NATPAGE and the PDF print form is done via naming conventions. For example, the FIELD control with the name "invoiceNumber" which is defined in the NATPAGE is bound to the corresponding CISFO control in the PDF print form, as shown below.

```
<natpage...
    ...
    <field valueprop="invoiceNumber" width="200">
    </field>
...
</natpage>
```

```
<cisfo:foppage2 ...
    ...
    <cisfo:celltext2 text="Sales Order:">
        <cisfo:textreplace2 valueprop="invoiceNumber">
        </cisfo:textreplace2>
    </cisfo:celltext2>
...
</cisfo:foppage2>
```

Built-in Events

When using the adapter listener `com.softwareag.cis.server.adapterlistener.PDFFOPListener`, the following events are supported and can be bound to a control's method property:

Event	Description
onPrintPDF	Creates a PDF document and opens it in a modal pop-up.
onUploadPDF	Uploads the PDF document to the Natural server. See also ***Documents*** in *Some Common Rules for all Controls*.

Advanced Data Binding and Rendering

Some advanced topics regarding the data binding between the properties defined in the NATPAGE page layout and the PDF print form are covered below.

- Displaying Field Lists
- Customizing Decimal Digits in Grids
- Customizing Boolean Texts in Grids
- Multi Language Settings with Boolean Texts
- Whitespace Handling

Displaying Field Lists

Example:

```
<njx:fieldlist fieldlistprop="myfieldlist" fieldcount="5" hdist="10">
    <njx:fielditem valueprop="myvalue" width="130" invisiblemode="invisible">
    </njx:fielditem>
</njx:fieldlist>
```

In the corresponding *_fop.xml* layout, the items of the above field list can be referenced as follows:

```
<cisfo:celltext2 text="First Item">
    <cisfo:textreplace2 valueprop="myfieldlist.items[0].myvalue">
    </cisfo:textreplace2>
</cisfo:celltext2>
...
<cisfo:celltext2 text="Second Item">
    <cisfo:textreplace2 valueprop="myfieldlist.items[1].myvalue">
    </cisfo:textreplace2>
</cisfo:celltext2>
```

Note that within the *_fop.xml* layouts, indexing starts with 0.

Customizing Decimal Digits in Grids

To customize the decimal digits for all items in a grid, you simply add an **XCIDATADEF** field and refer to this central field as shown in the example below:

```
<xcidatadef dataprop="myDecimalDigits" datatype="int">
</xcidatadef>
```

```
<cisfo:repeat2>
  <cisfo:row2>
    <cisfo:replace2 valueprop="unitPrice" datatype="float"
                    decimaldigitsprop="/myDecimalDigits" bordercolor="black"
                    borderstyle="solid" borderwidth="1pt" padding="2pt"
                    textalign="right">
    </cisfo:replace2>
    <cisfo:replace2 valueprop="price" datatype="float"
                    decimaldigitsprop="/myDecimalDigits" bordercolor="black"
                    borderstyle="solid" borderwidth="1pt" padding="2pt"
                    textalign="right">
    </cisfo:replace2>
  </cisfo:row2>
</cisfo:repeat2>
```

If you would like to customize the decimal digits per item in a grid, add a corresponding XCIDATADEF field as a grid column as shown in the example below:

```
<textgridsss2 griddataprop="orderedProducts" rowcount="3" width="100%">
    <column name="Unit Price" property="unitPrice" width="100">
    </column>
    <column name="Price" property="price" width="100">
    </column>
    <xcidatadef dataprop="itemDecimalDigits" datatype="int">
    </xcidatadef>
</textgridsss2>
```

The name "itemDecimalDigits" is not fixed. You can choose any valid property name. From within the corresponding **_fop.xml* layout, you can refer to this setting as follows:

```
<cisfo:repeat2>
  <cisfo:row2>
    <cisfo:replace2 valueprop="unitPrice" datatype="float"
                    decimaldigitsprop="itemDecimalDigits" bordercolor="black"
                    borderstyle="solid" borderwidth="1pt" padding="2pt"
                    textalign="right">
    </cisfo:replace2>
    <cisfo:replace2 valueprop="price" datatype="float"
                    decimaldigitsprop="itemDecimalDigits" bordercolor="black"
                    borderstyle="solid" borderwidth="1pt" padding="2pt"
                    textalign="right">
    </cisfo:replace2>
```

```
  </cisfo:row2>
</cisfo:repeat2>
```

Customizing Boolean Texts in Grids

To customize the Boolean texts for all items in a grid, you simply add an XCIDATADEF field and refer to this central field as shown in the example below:

```
<xcidatadef dataprop="mybooleantexts">
</xcidatadef>
```

```
<cisfo:repeat2>
  <cisfo:row2>
    <cisfo:replace2 valueprop="myChoice" datatype="boolean"
                    booleantextprop="/mybooleantexts" bordercolor="black"
                    borderstyle="solid" borderwidth="1pt" padding="2pt"
                    textalign="right">
    </cisfo:replace2>
    ...
  </cisfo:row2>
</cisfo:repeat2>
```

If you would like to customize the Boolean texts per item in a grid, add a corresponding XCIDATADEF field as a grid column as shown in the example below:

```
<textgridsss2 griddataprop="orderedProducts" rowcount="3" width="100%">
    <column name="Choice" property="myChoice" datatype="xs:boolean" width="100">
    </column>
    ...
    <xcidatadef dataprop="itemBooleanTexts" >
    </xcidatadef>
</textgridsss2>
```

The name "itemBooleanTexts" is not fixed. You can choose any valid property name. From within the corresponding **_fop.xml* layout, you can refer to this setting as follows:

```
<cisfo:repeat2>
  <cisfo:row2>
    <cisfo:replace2 valueprop="myChoice" datatype="boolean"
                    booleantextprop="itemBooleanTexts" bordercolor="black"
                    borderstyle="solid" borderwidth="1pt" padding="2pt"
                    textalign="right">
    </cisfo:replace2>
    ...
  </cisfo:row2>
</cisfo:repeat2>
```

Multi Language Settings with Boolean Texts

`textid` properties are currently not available for Boolean texts. However, you can set the text value in the Natural application depending on the language. Example:

```
IF *LANGUAGE = 2
THEN
   MYBOOLEANVALUES:="Ja;Nein;"
ELSE
   MYBOOLEANVALUES:="Yes;No;"
END-IF
```

Whitespace Handling

In most cases you choose specific `cisfo:*` controls and containers to reflect the intended whitespaces like line-breaks, tabs and spaces in your reports. However, in some cases you may have the need to simply apply existing text with whitespaces as values of `cisfo:*` controls or the `REPORT` and `REPORT2` control data fields. If you do this, per default spaces and tabs are collapsed. This means, for instance, five spaces are collapsed into one space, and a line-break or a tab is simply rendered as one space. You can customize this behavior for the following controls:

- CISFO:TEXT2
- CISFO:REPLACE2
- CISFO:TEXTREPLACE2
- CISFO:STYLE2

These controls support two properties: `whitespacepreserve` and `tabspaces` (see chapter *Application Designer > PDF and FOP Services*).

You also can customize this behavior for `TEXT` fields in the `REPORT` and `REPORT2` control data structure. Set the property `whitespacehandling` to *true* if you want to customize whitespace handling for specific report text fields. This will generate two additional fields - WHITESPACEP-RESERVE and TABSPACES in the Natural data structure.

You'll find corresponding samples in the NaturalAjaxDemos.

XI Working with Icons

114 Working with Icons

- Using Own Icons 862
- Using Bootstrap Icons 862
- Using Bootstrap Icons as Icon Fonts 862
- Using Bootstrap Icons as SVG Icon Files 864
- Using SVG from Natural Programs 864

Using Own Icons

You can use your own icons in controls and in CSS in the following ways:

- Reference icon files by URL, whereby the icons do not have to be contained in the User Interface Component or in the web application
- Copy icon files to the User Interface Component and use the relative URL

Using Bootstrap Icons

The packaged Bootstrap Icons (**https://icons.getbootstrap.com/**) can be used in controls and CSS. The Bootstrap Icons library contains over 2000 icons. The icons are available as icon fonts as well as SVG icons. The icon library comes along within the cisnatural.war in the following subfolder:

cisnatural/HTMLBasedGUI/bootstrapicons

The subfolder */icons* contains the SVG icon files.

To see the available icons, open the url *https://icons.getbootstrap.com/sprite/*

Using Bootstrap Icons as Icon Fonts

Icon fonts can be used in responsive and non-responsive controls. The required CSS file is automatically integrated into the page. There is no need to copy files to the User Interface Component. No additional files need to be applied to the pages.

Support in Responsive Controls

Icon fonts can be used in name/nameprop properties of all controls. A BMOBILE:BUTTON control with a name value

```
BUTTON <i class="bi-person-square"></i>
```

will be rendered as



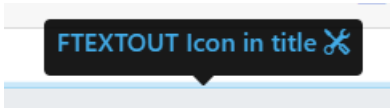
The color of the icon can be manipulated by applying CSS style setting:

```
BUTTON <i style=”color:green” class="bi-person-square"></i>
```

Icon fonts can also be used in the title/titleprop properties (tooltips). A BMOBILE:FTEXTOUT with a title value

```
FTEXTOUT Icon in title <i class="bi-tools"></i>
```

Will be rendered as:



The BMOBILE:IHTML also support this.

Support in Non-Responsive Controls

Icon fonts are supported in the following controls:

COLUMN
CSVCOLUMN
METHODLINK
STATUSBAR
TEXTOUT
TITLEBAR

The IHTML also supports this feature.

However, icon fonts are not supported in title/titleprop.

Tools support in NaturalONE

You can now conveniently search and select an icon font and copy it to your layout XML or to your Natural program through Bootstrap Icons in the Ajax Developer menu.

Using Bootstrap Icons as SVG Icon Files

SVG icon files can be used from the library without copying any file to the User Interface Component. It is sufficient to set a relative URL property value.

Example:

```
<bmobile:icon iconurl="../HTMLBasedGUI/bootstrapicons/icons/person-circle.svg">
```

Will be rendered as



Using SVG from Natural Programs

SVG XML can be applied as property value from Natural program. In responsive controls this is supported in all controls with `nameprop` and `titleprop`.

IHTML and BMOBILE:IHTML also support this feature.

However, in non-responsive controls SVG XML is only supported in controls that also support a `straighttext` property.

Example:

```
<svg class='bi' width='128' height='128' fill='currentColor'>
<use href='../HTMLBasedGUI/bootstrapicons/bootstrap-icons.svg#shop'/>
</svg>
```

in an `BMOBILE:IHTML` control will be rendered as:

