

Natural for Mainframes

Natural for Ajax

Version 4.2.6 for Mainframes

October 2009

This document applies to Natural Version 4.2.6 for Mainframes and to all subsequent releases.

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

Copyright © Software AG 1979-2009. All rights reserved.

The name Software AG, webMethods and all Software AG product names are either trademarks or registered trademarks of Software AG and/or Software AG USA, Inc. Other company and product names mentioned herein may be trademarks of their respective owners.

Table of Contents

1 Natural for Ajax	1
2 Introduction	3
What is a Rich Internet Application?	4
Rich Internet Applications with Natural	4
Mixed Applications	5
3 Installation	7
Prerequisites	8
License Key File Handling	11
Installing Natural for Ajax on JBoss Application Server	11
Installing Natural for Ajax on Sun Java System Application Server	15
Verifying the Installation	18
4 Setting Up Your Environment	19
Setting Up Application Designer	20
Setting Up Your Development Environment for Natural	20
Setting Up Your Runtime Environment for Natural	21
5 First Steps	25
6 About this Tutorial	27
7 Starting the Development Workplace	31
8 Creating a Project	33
9 Getting Started with the Layout Painter	35
Creating a New Layout	36
Elements of the Layout Painter Screen	38
Previewing the Layout	39
Viewing the XML Code	40
10 Writing the GUI Layout	43
Specifying the Properties for the Natural Page	44
Specifying a Name for the Title Bar	45
Using the Property Editor	46
Specifying a Name and Method for the Button	48
Adding the Input and Output Areas	48
Adding the Image	52
Adding a Horizontal Distance	52
Adding an Instructional Text	53
Adding a Vertical Distance	54
Saving Your Layout	54
11 Setting Up Your Development and Runtime Environment for Natural	57
12 Creating the Natural Code	59
Importing the Adapter into Natural	60
Creating the Main Program	61
Testing the Completed Application	64
13 Some Background Information	67
Name Binding between Controls and Adapter	68
Data Exchange at Runtime	68

Files and their Locations	69
14 Developing the User Interface	71
Starting the Development Workplace	72
Creating an Application Designer Project	73
Creating a Natural Page	73
Specifying Properties for the Natural Page	74
Designing the Page	75
Binding Properties and Methods	76
Previewing the Layout	76
Viewing the Protocol	77
Saving the Layout	77
Generating the Adapter	77
Data Type Mapping	78
15 Developing the Application Code	79
Importing the Adapter	80
Creating the Main Program	84
Structure of the Main Program	86
Handling Page Events	86
Built-in Events and User-defined Events	87
Sending Events to the User Interface	87
Using Pop-Up Windows	88
Using Natural Maps	90
Navigating between Pages and Maps	90
Using Pages and Maps Alternatively	91
Starting a Natural Application from the Logon Page	92
Starting a Natural Application with a URL	92
16 Deploying the Application	93
Components of a Natural for Ajax Application	94
Unloading Natural Modules	94
Unloading the User Interface Components	94
Installing the Natural Modules	95
Installing the User Interface Components	95
17 Natural Parameters and System Variables	97
18 Usage of Edit Masks	99
General Information	100
Data Types with Edit Masks	100
Characters Used in Edit Masks	100
Specifying Edit Masks in Layouts	101
Static versus Dynamic Validation	103
19 Multi Language Management	105
20 Support of Right-to-Left Languages	107
21 Server-Side Scrolling and Sorting	109
General Information	110
Variants of Server-Side Scrolling and Sorting	110
Controls that Support Server-Side Scrolling and Sorting	114

Data Structures for Server-Side Scrolling and Sorting	114
Server-Side Scrolling and Sorting in Trees	116
Events for Server-Side Scrolling and Sorting	117
22 Application Modernization	119
23 Overview of Conversion Steps	121
24 Map Extraction	123
General Information	124
Using Natural for Ajax Tools	124
Using the Mass Function	124
Location of the Files	124
25 Map Conversion	127
General Information	128
First Steps	129
Using the Map Converter	137
Using the Editor Extension	141
Using the Conversion Rules Tool	144
Using the Conversion Logs Tool	146
26 Customizing the Map Conversion Process	149
Map Converter Processing	150
Conversion Rules	154
Templates	164
Tag Converters	167
27 Code Conversion	169
General Information	170
Generating Adapters	170
Structure of a Map-Based Application	170
Structure of a Natural for Ajax Application	171
Tasks of the Code Conversion	172
DEFINE DATA Statement	172
INPUT Statement	173
REINPUT Statement	174
PF-Key Event Handling	176
SET KEY Statement	177
Processing Rules	181
System Variables	181
Variable Names Containing Special Characters	182
28 Working with Containers	185
29 Positioning of Controls inside a Container	187
Row Types - TR and ITR	188
Some More Details on ITR	189
TR Properties	191
ITR Properties	192
30 Defining the Width of Controls inside a Container	195
Controlling the Width of Controls	196
HDIST and VDIST Controls	198

HDIST Properties	201
VDIST Properties	201
rowspan and colspan Definitions	202
CELLSPAN Control	203
CELLSPAN Properties	204
Rules for Positioning Controls inside Containers	206
31 Vertical Sizing of Containers and Controls	207
Vertical Pixel Sizing	208
Vertical Percentage Sizing	209
Finishing the Example	212
32 Overview of Different Containers	215
Different Kind of Containers	216
Row Containers	216
Column Containers	217
Row and Column Containers in Combination	218
Nesting Containers	219
33 ROWAREA and COLAREA	221
ROWAREA Properties	222
COLAREA Properties	229
34 ROWAREAWITHHEADER	235
Simple Example	236
ROWAREAWITHHEADER Properties	237
ROWAREAHEADER Properties	239
ROWAREABODY Properties	240
35 ROWTABAREA and COLTABAREA	243
ROWTABAREA Properties	245
COLTABAREA Properties	258
TABPAGE Properties	588
The Most Common Error	267
Example: Controlling which Tab is displayed by the Server Adapter	267
Example: Controlling the Visibility of Tab Pages	268
36 ROWTABLE0 and COLTABLE0	271
ROWTABLE0 Properties	273
COLTABLE0 Properties	274
37 ROWDYNAVIS and COLDYNAVIS	277
ROWDYNAVIS Properties	279
COLDYNAVIS Properties	280
Some Comments on Controlling the Visibility of Controls	282
38 ROWDIV and INNERDIV	283
When to Use ROWDIV and INNERDIV Containers	286
ROWDIV Properties	287
INNERDIV Properties	288
39 ROWSCROLLAREA	291
ROWSCROLLAREA Properties	293
40 HSPLIT and VSPLIT	297

Example for HSPLIT	298
Example for VSPLIT	300
HSPLIT Properties	301
VSPLIT Properties	302
SPLITCELL Properties	303
Defining the Split Size	304
41 HLINE and VLINE	305
VLINE Properties	307
HLINE Properties	308
42 Performance Optimization with Containers	309
43 Working with Controls	313
44 Some Common Rules for all Controls	317
Name and Text ID	318
Table, Row, Column, Control	318
Explicit Alignment	318
Binding to Adapter Parameters	319
Directly Influencing the Control Style	319
Dynamically Controlling the Visibility and the Display Status of Controls	320
Focus Management	321
Flushing of Inputs	322
Tab Sequence	322
Tooltips	324
45 BREADCRUMB	325
Example	326
Adapter Interface	326
Built-in Events	326
Properties	327
46 BUTTON	329
Example: Simple Button	330
Example: Button with Image	331
Hiding and Disabling Buttons	331
Properties	331
47 BUTTONLIST	337
Adapter Interface	338
Properties	338
48 CHECKBOX	341
Properties	342
49 COMBODYN2	347
Adapter Interface	348
Properties	348
50 COMBOFIX	353
COMBOFIX Properties	354
COMBOOPTION Properties	357
51 DATEINPUT	359
Example	360

Properties	360
52 DROPICON	367
Example	368
Properties	368
53 FIELD	373
Built-in Events	374
Properties	374
54 FILEUPLOAD/FILEUPLOAD2	385
FILEUPLOAD	386
FILEUPLOAD2	388
FILEUPLOAD Properties	389
FILEUPLOAD2 Properties	392
55 ICON	395
Example	396
Properties	396
56 ICONLIST	401
Adapter Interface	402
Built-in Events	402
Properties	402
57 IHTML	405
Properties	406
58 IMAGEOUT	409
Properties	410
59 LABEL	413
Example	415
Aligning the Text	415
Properties	416
60 MENUBUTTON	421
Example	422
MENUBUTTON Properties	423
MENUITEM Properties	425
61 METHODLINK	427
Properties	428
62 MULTISELECT	433
Example	434
Adapter Interface	434
Properties	434
63 NEWSFEED	439
Example	441
Built-in Events	442
Properties	442
64 RADIOBUTTON	443
Properties	444
65 SCHEDULELINE	449
Properties	450

66 SLIDER	455
Example	456
Adapter Interface	457
Properties	457
67 STRIPSEL	463
Example	464
Properties	464
68 SUBPAGE	469
Properties	470
69 TABSEL	473
Adapter Interface	474
Built-in Events	475
Properties	475
70 TABSTRIP2	477
Example	478
Adapter Interface	478
Built-in Events	478
Properties	479
71 TAGCLOUD	481
Example	482
Adapter Interface	483
Built-in Events	483
Properties	483
72 TEXT	487
Properties	488
73 TEXTOUT	495
Example	496
Properties	496
74 TOGGLE	503
Properties	504
75 ACTIVEX	509
Properties	510
76 GOOGLEMAP2	513
Before You Start	514
Example	515
Typical Problems	516
Properties	517
77 NETMEETING	519
Example	520
Properties	520
78 SKYPECALL	523
Example	525
Properties	525
79 NJX:BUTTONITEMLIST	527
Example	529

Adapter Interface	529
Built-in Events	530
Properties	530
80 NJX:BUTTONITEM	531
Example	532
Built-in Events	532
Properties	533
81 NJX:BUTTONITEMLISTFIX	537
Example	538
Adapter Interface	538
Built-in Events	539
Properties	539
82 NJX:BUTTONITEMFIX	541
Example	542
Built-in Events	542
Properties	543
83 NJX:FIELDLIST	549
Example	551
Adapter Interface	552
Built-in Events	552
Properties	552
84 NJX:FIELDITEM	555
Example	557
Adapter Interface	558
Built-in Events	558
Properties	558
85 NJX:FIELDVALUE	569
Example	571
Adapter Interface	571
Built-in Events	571
Properties	572
86 NJX:NJXVARIABLE	581
Example	582
Properties	582
87 NJX:EVENTDATA	583
Example	585
Adapter Interface	586
88 NATPAGE	587
Properties	588
89 Working with Grids	595
90 Basics	597
91 TEXTGRID2	599
A Simple Example	600
Adapter Interface	601
Selecting Rows in a TEXTGRID2	601













TEXTGRID2 Properties	602
COLUMN Properties	608
Dynamic Setting of Text Styles in TEXTGRID2	612
92 TEXTGRIDSSS2 - TEXTGRID2 with Server-Side Scrolling	613
Performance Considerations	614
Example	614
Adapter Interface	616
Using Server-Side Scrolling	616
Using Server-Side Sorting	617
TEXTGRIDSSS2 Properties	617
93 ROWTABLEAREA2 - The Flexible Control Grid	625
Example	626
Adapter Interface	628
Built-in Events	628
Making Grids Look like Grids	629
ROWTABLEAREA2 Properties	630
STR Properties	636
94 MGDGRID - Managing the Grid	639
Example	641
Adapter Interface	642
Built-in Events	643
MGDGRID Properties	643
ROWINSERT Properties	647
ROWCOPY Properties	648
ROWDELETE Properties	649
95 GRIDCOLHEADER - Flexible Column Headers	651
Flexible Column Sizing	652
Flexible Column Sorting	655
GRIDCOLHEADER Properties	656
Smart Selection of Rows - SELECTOR Control	659
SELECTOR Properties	660
96 Working with Trees	663
97 Basics	665
Types of Trees	666
When to Use Which Type	667
98 TREENODE3 in Control Grid (ROWTABLEAREA2)	669
Example	670
Adapter Interface	671
Built-in Events	671
Properties	671
99 CLIENTTREE	677
Example	678
Adapter Interface	679
Built-in Events	679
Properties	679









100 Working with Menus	683
101 Types of Menus	685
102 MENU	687
Example	688
Adapter Interface	689
Built-in Events	689
Properties	690
103 DLMENU	693
Example	694
Adapter Interface	695
Built-in Events	695
Properties	696
104 XCIPOPUPMENU - Enable Context Menus	697
Example	698
Adapter Interface	699
Built-in Events	700
105 Non-Visual Controls and Hot Keys	701
106 TIMER	703
Example	704
Properties	705
107 XCIDATADEF - Data Definition	707
Example	708
Properties	711
108 Extended Hot Key Management	713
Direct Hot Key Definitions with Certain Controls	714
Hot Key Definitions for Certain Controls	714
109 Function Key Handling	717
110 Working with Workplaces	719
111 What are Multi Frame Pages?	721
112 Definition of Multi Frame Pages	723
MFPAGE	724
MFCISFRAME	726
MFHTMLFRAME	728
MFFRAMESET	729
113 Application Designer Workplace Framework	733
Framework Overview	735
Functions Frame: MFWPFUNCTIONS	736
Active Functions Frame: MFWPACTIVEFUNCTIONS	738
Content Frame: MFWPCONTENT	739
Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO	741
Session Management inside the Workplace	750
Workplace API for Dynamic Manipulation	751
114 NJX:XCIWPINFO2	753
Example	754
Adapter Interface	754

115 NJX:XCIWPFUNCTIONS	759
Example	760
Adapter Interface	760
116 NJX:XCIACCESS2	765
Example	766
Adapter Interface	766
Index	769

1 Natural for Ajax

This documentation is organized under the following headings:

Using Natural for Ajax		
	Introduction	What is Natural for Ajax?
	Installation	How to install Natural for Ajax on the supported application servers.
	Setting Up Your Environment	How to set up Application Designer, your development environment for Natural, and your runtime environment for Natural.
	First Steps	How to create a “Hello World!” application.
	Developing the User Interface	How to develop the user interface using Application Designer.
	Developing the Application Code	How to develop the application code using Natural Studio or Natural for Eclipse.
	Deploying the Application	How to unload and install the Natural modules and user interface components.
	Natural Parameters and System Variables	Gives an overview of the Natural parameters and system variables that are evaluated in Natural for Ajax applications and sent to Application Designer.
	Usage of Edit Masks	Describes how Natural for Ajax supports the Natural edit mask concept.
	Multi Language Management	Describes aspects to be considered for internationalization.
	Support of Right-to-Left Languages	Describes how Natural for Ajax supports right-to-left languages and bidirectional text.
	Server-Side Scrolling and Sorting	Describes how Natural for Ajax supports the concept of server-side scrolling and sorting.

	Application Modernization	How to convert a character-based Natural application to a Natural for Ajax application.
Application Designer Reference (adapted to Natural for Ajax)		
	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 Workplaces	Deals with applications that organize multiple pages in so-called workplaces.

See also *Configuring the Client* in the *Natural Web I/O Interface* documentation. There, you will learn how to

- start a Natural application from the logon page or with a URL,
- manage the configuration file for the session using the configuration tool,
- modify the style sheet which controls the font, the color and the representation of the PF keys,
- activate the preconfigured security settings of Natural for Ajax and to adapt them to your requirements,
- create your own trust files for a secure connection between the Natural Web I/O Interface server and Natural for Ajax,
- enable logging in the case of problems with Natural for Ajax.

2 Introduction

■ What is a Rich Internet Application?	4
■ Rich Internet Applications with Natural	4
■ Mixed Applications	5

Using Natural for Ajax, you can create rich internet applications which use the Ajax (Asynchronous JavaScript and XML) technology. This enables Natural users on Windows, UNIX and mainframe platforms to develop and use Natural applications with a browser-based user interface, similar to GUI desktop applications.

What is a Rich Internet Application?

Classical HTML- and browser-based applications suffer from known disadvantages. The server responds to each user interaction with a new page. This may lead to long response times and new rendering in the browser and thus to a discontinuous workflow for the user. The possibilities offered by DHTML overcome these disadvantages, but they are complicated to use and make it hard to build a comfortable user interface. The user interface is therefore often simpler and less comfortable than users are accustomed to from their experience with desktop applications. Although it is possible to provide complex controls and features like drag-and-drop, this is hard to implement - especially if compatibility with all commonly used browsers is required. Classical GUI applications also have the disadvantage that a client component of the application must be installed on each client machine.

Rich internet applications that use the Ajax technology overcome these disadvantages by combining the reachability of browser-based applications with the rich user interface of GUI applications. Software AG provides support for the development of rich internet applications with Application Designer. Natural for Ajax combines the user interface capabilities of Application Designer with the application development capabilities of Natural.

Rich Internet Applications with Natural

At runtime, a rich internet application with Natural has the following structure:

- A Natural host session on a Windows, UNIX or mainframe server runs the application code. Other than with a map application, the application does not deal with user interface issues. It contains only the application logic and communicates with the user interface layer by sending and receiving data. The data is displayed in page in a web browser. Events - such as button clicks - that the user raises in the web browser are passed back to the application code. Along with an event, the application code receives also the data that the user modified in the web browser. It processes the event and the data and returns modified data back to the web browser page.
- Natural for Ajax, which is running on an application server, merges the data received from the Natural application into a DHTML page and delivers the page to the web browser. In the inverse direction, Natural for Ajax forwards events that the user raised in the web browser along with the modified data to the Natural application.

- A web browser renders the DHTML page. JavaScript code on the page processes local user interaction and exchanges data with Natural for Ajax as needed. It uses Ajax technology to exchange data with the Natural application in the background without having to re-render the page as a whole.

At development time, a rich internet application is created with Natural in the following way:

- Application Designer is used to develop the user interface layout of a web page and to bind the controls on the page to data elements in the application. Application Designer is contained in the Natural for Ajax module running on the application server.
- When the user saves the page layout, a Natural module of type “Adapter” is generated. The adapter serves as an interface between the application code and the page layout. It contains:
 - A data structure that describes the data that the Natural application has to deliver to the application server in order to populate the web page.
 - The Natural code necessary to transfer the data structure to the user interface and to receive modified data back.
 - A code skeleton, in the form of comment lines, that contains handlers for the expected events. The application programmer can copy this code skeleton into the main program to implement the event handlers.
- Then a main program is implemented that exchanges data with the web page using the adapter and handles the events. The event handler code has no knowledge of the web page layout, but operates only on the page data that is sent and received through the adapter.
- The navigation between different pages is implemented. A rich internet application navigates between pages in the same way as a map application would navigate between maps.

Mixed Applications

With the support of Unicode, Natural has introduced the Natural Web I/O Interface which renders Natural maps in a web browser. Typically, if you are running map-oriented applications and wish to change them to rich internet applications, you will do this gradually. In certain parts of an application, maps might be replaced by rich GUI pages, other parts will possibly be left unchanged. Therefore, Natural supports running mixed applications which consist of both maps and rich GUI pages. With maps, the application controls the page layout, and the rendering mechanism therefore respects the layout information that the application provides. With rich GUI pages, the application does not control the layout; the layout is controlled by Application Designer. However, for the users of an application the switch between maps and rich GUI pages is seamless.

3

Installation

■ Prerequisites	8
■ License Key File Handling	11
■ Installing Natural for Ajax on JBoss Application Server	11
■ Installing Natural for Ajax on Sun Java System Application Server	15
■ Verifying the Installation	18

Natural for Ajax consists of a J2EE enterprise application (*njx12.ear*) and a J2EE resource adapter (*njx12ra.rar*). Both components are to be deployed on a J2EE server. Natural for Ajax receives data from Natural applications running on a Windows, UNIX or mainframe host and delivers web pages to the user's web browser.

This chapter describes the installation of Natural for Ajax on application servers on Windows or UNIX. It does not describe the installation of the additionally required Natural components on a Windows, UNIX or mainframe host, but refers to the corresponding installation documents.

For information on how to activate the preconfigured security settings of Natural for Ajax and how to adapt them to your requirements, see *Configuring Security* in the *Configuring the Client* part of the *Natural Web I/O Interface* documentation.

Prerequisites

The following topics are covered below:

- [Java](#)
- [J2EE Server](#)
- [Apache Ant](#)
- [Natural for Mainframes](#)
- [Natural for UNIX](#)
- [Natural for Windows](#)
- [Support for Special Features](#)
- [Development Servers](#)
- [Development Clients](#)
- [Browser Prerequisites](#)

Java

JDK 1.5.0_12 or above is required.

J2EE Server

The following application servers are supported. The application servers are not delivered with Natural for Ajax. They can be obtained from the locations indicated below, according to their respective license terms.

- JBoss Application Server 4.0.5 and 4.2.2 (see <http://www.jboss.org/>).
- Sun Java System Application Server 8.1, 8.2 and 9.1 (see <http://www.sun.com/>).

Apache Ant

Apache Ant 1.6.5 or above is required to perform the deployment on JBoss Application Server. This tool is freely available on <http://ant.apache.org/>.

Natural for Mainframes

If you want to use Natural for Ajax with Natural for Mainframes, the following must be installed:

- Natural for Mainframes Version 4.2.3 or above, and
- the Natural Web I/O Interface server.

For detailed information, see:

- the *Installation* documentation which is provided with Natural for Mainframes;
- the section *Installing and Configuring the Natural Web I/O Interface Server* in the version of the *Natural Web I/O Interface* documentation which is provided for Natural for Mainframes.

Natural for UNIX

If you want to use Natural for Ajax with Natural for UNIX, the following must be installed:

- Natural for UNIX Version 6.3.1 or above, and
- the Natural Web I/O Interface daemon.

For detailed information, see:

- the *Installation* documentation which is provided for Natural for UNIX;
- the section *Installing and Configuring the Natural Web I/O Interface Server* in the version of the *Natural Web I/O Interface* documentation which is provided for Natural for UNIX.

Natural for Windows

If you want to use Natural for Ajax with Natural for Windows, the following must be installed:

- Natural for Windows Version 6.3.3 or above, and
- the Natural Web I/O Interface server (which is implemented as a service).

For detailed information, see:

- the *Installation* documentation which is provided for Natural for Windows;
- the section *Installing and Configuring the Natural Web I/O Interface Server* in the version of the *Natural Web I/O Interface* documentation which is provided for Natural for Windows.

Support for Special Features

If you want to use the Natural parameters `DC` and `DTFORM` in a Natural for Ajax application, the following versions are required:

- Natural for Mainframes Version 4.2.5 or above,
- Natural for UNIX Version 6.3.5 or above,
- Natural for Windows Version 6.3.5 or above.

If you want to use the workplace example mentioned in *Application Designer Workplace Framework*, the following versions are required:

- Natural for Mainframes Version 4.2.6 or above,
- Natural for UNIX Version 6.3.6 or above,
- Natural for Windows Version 6.3.6 or above.

If you want to use the example on control variables mentioned in *XCIDATADEF - Data Definition* or the Natural system variable `*CURS-FIELD` in a Natural for Ajax application, the following versions are required:

- Natural for Mainframes Version 4.2.6 or above,
- Natural for UNIX Version 6.3.7 or above,
- Natural for Windows Version 6.3.7 or above.

Development Servers

The following development servers support the remote development of Natural for Ajax applications:

- Natural Development Server for Mainframes Version 2.2.3 or above.
- Natural Development Server for UNIX Version 2.2.3 or above.
- Natural Development Server for Windows Version 2.2.4 or above.

Development Clients

The following development clients support the remote development of Natural for Ajax applications:

- Natural for Windows (Natural Studio) Version 6.3.1 or above.
- Natural for Eclipse Version 3.2.1 or above.

Browser Prerequisites

Supported browsers in this version are:

- Internet Explorer 6.0 through 7.0.
- Mozilla Firefox 2.0. through 3.0.



Important: Cookies and JavaScript must be enabled in the browser.

License Key File Handling

A valid license key file is required during the installation. The license key file is an XML file which is usually supplied along with the product. Alternatively, you can obtain a license key file from Software AG via your local distributor.

Installing Natural for Ajax on JBoss Application Server

Only one version of the Natural Web I/O Interface client or one version of Natural for Ajax can be installed on the same JBoss Application Server.

You can either install the Natural Web I/O Interface client or Natural for Ajax on the same JBoss Application Server, not both.

It is assumed that `<jboss>` is the directory of your JBoss Application Server installation.

The following topics are covered below:

- [First-time Installation](#)
- [Update Installation](#)

First-time Installation

► To install Natural for Ajax

- 1 Install Apache Ant (you need Apache Ant to deploy Natural for Ajax to the JBoss Application Server; see the [Prerequisites](#) above for the required version number):
 1. Download and unzip Apache Ant (from <http://ant.apache.org/>) into an installation directory of your choice. Avoid a directory name that contains blanks.
 2. Let the environment variable `ANT_HOME` point to the directory `<ant>` (where `<ant>` is the directory of your Ant installation).

3. Add `<ant>/bin` to your `PATH` environment variable.
- 2 Deploy Natural for Ajax to JBoss Application Server:
 1. Copy the Natural for Ajax distributables to a directory on a disk drive.
 2. In the directory that contains the Natural for Ajax distributables, there is an Ant script named `jbossdeploy.xml`. Edit this script and change the setting

```
<property name="jbossHome" value=""/>
```

to

```
<property name="jbossHome" value="<jboss>"/>
```

where `<jboss>` is your JBoss Application Server installation directory.



Important: Take care to use forward slashes (also on Windows) when specifying the directory path.

3. Execute the script `jbossdeploy.xml` by entering the following command:

```
ant -f jbossdeploy.xml
```

Wait for the message “BUILD SUCCESSFUL”. This indicates that the deployment was successful.

- 3 Copy the license file into the directory `<jboss>/server/default/deploy/njx12.ear/cisnatural.war/cis/licensekey`.
- 4 Edit the file `<jboss>/server/default/deploy/jbossjca-service.xml` and change the setting

```
<!-- Enable connection close debug monitoring -->  
<attribute name="Debug">true</attribute>
```

to

```
<!-- Enable connection close debug monitoring -->  
<attribute name="Debug">false</attribute>
```

- 5 JBoss Application Server 4.0.5 only: Edit the file `<jboss>/server/default/deploy/njx12.ear/cisnatural.war/WEB-INF/web.xml` and uncomment the section

```
<!--
Uncomment the next lines, in case the configuration tool is installed on a JBOSS
4.0.5.GA
<listener>

<listener-class>org.apache.myfaces.webapp.StartupServletContextListener</listener-class>
</listener>
-->
```

so that it looks as follows:

```
<listener>

<listener-class>org.apache.myfaces.webapp.StartupServletContextListener</listener-class>
</listener>
```



Important: For JBoss Application Server 4.2, you must not remove this comment.

- 6 Start JBoss Application Server.

Update Installation

► To update Natural for Ajax

- 1 Shut down JBoss Application Server.
- 2 Deploy Natural for Ajax to JBoss Application Server:
 1. Copy the Natural for Ajax distributables to a directory on a disk drive.
 2. In the directory that contains the Natural for Ajax distributables, there is an Ant script named `jbossdeploy.xml`. Edit this script and change the setting

```
<property name="jbossHome" value="" />
```

to

```
<property name="jbossHOME" value="<jboss>"/>
```

where *<jboss>* is your JBoss Application Server installation directory.



Important: Take care to use forward slashes (also on Windows) when specifying the directory path.

3. In order to upgrade an existing Natural for Ajax 1.1.1 installation to version 1.2, execute the script *jbossdeploy.xml* by entering the following command:

```
ant -f jbossdeploy.xml upgrade
```

Wait for the message “BUILD SUCCESSFUL”. This indicates that the deployment was successful.

4. In order to update an existing Natural for Ajax 1.2.<*n*> installation to the newest update package (1.2.<*m*>), execute the script *jbossdeploy.xml* by entering the following command:

```
ant -f jbossdeploy.xml redeploy
```

Wait for the message “BUILD SUCCESSFUL”. This indicates that the deployment was successful.

5. The update installation creates two backup directories, *backup* and *backupsys*, in the JBoss Application Server installation directory. These directories contain a backup of the user project directories and some configuration files. Should the update installation fail for some reason, these directories can be used to restore the the backed up user data in a fresh installation. After a successful update installation, these directories can be removed.

- 3 Make sure that the file *<jboss>/server/default/deploy/jbossjca-service.xml* contains the same settings as described for a first-time installation.
- 4 JBoss Application Server 4.0.5 only: Make sure that the file *<jboss>/server/default/deploy/njx12.ear/cisnatural.war/WEB-INF/web.xml* contains the same settings as described for a first-time installation.
- 5 Regenerate the HTML pages of the projects that you have created with an earlier release of Natural for Ajax. For each project to regenerate, execute the script *jbossdeploy.xml* by entering the following command:

```
ant -f jbossdeploy.xml regenerate -Dnjxproj=<projectname>
```

Wait for the message “BUILD SUCCESSFUL”. This indicates that the generation was successful.

- 6 Start JBoss Application Server.

Installing Natural for Ajax on Sun Java System Application Server

Natural for Ajax is installed using the Administration Console of Sun Java System Application Server.

The following is assumed:

- *<host>* is the name of the machine on which the application server is installed.
- *<port>* is the name of the port where the application server is installed. In a default installation, this is port 8080.
- *<adminport>* is the name of the port where the Administration Console is installed. In a default installation, this is port 4848.
- *<sunas>* is the path to the directory in which the application server is installed. In a default installation on Windows, this is *C:/Sun/AppServer*.

The following topics are covered below:

- [First-time Installation](#)
- [Update Installation](#)

First-time Installation

► To install Natural for Ajax

- 1 Edit the file *<sunas>/domains/domain1/config/server.policy* and add the following settings:

```
// Allow Application Designer to create an own class loader
grant {
  permission java.lang.RuntimePermission "createClassLoader";
};

// Allow Application Designer to modify its own project directories
grant {
  permission java.io.FilePermission
    "${com.sun.aas.instanceRoot}${/}applications${/}j2ee-apps${/}nrx12${/}cisenatural_war${/}-",
    "read,write,delete";
};

// Enable the Java Logging API
grant {
  permission java.util.logging.LoggingPermission "control";
};
```



Important: If you do not enable the Java Logging API, the resource adapter will not start and Natural for Ajax will therefore be inoperative.

- 2 Start the application server.
- 3 Open your web browser and enter the following URL:

```
http://<host>:<adminport>
```

This opens the Administration Console.

- 4 Deploy the resource adapter *njx12ra.rar*:
 1. Open the tree node **Applications > Connector Modules**.
 2. Choose **Deploy**.
 3. Select *njx12ra.rar* as the package file to be uploaded to the application server.
 4. Choose **Next**. "njx12ra" is automatically included as the application name.
 5. Choose **Finish**.
- 5 Define the JNDI name for the resource adapter:
 1. Open the tree node **Resources > Connectors > Connector Connection Pools**.
 2. Choose **New**.
 3. Enter "NatPool" (the name is arbitrary) as the name.
 4. Select **njx12ra** as the resource adapter.
 5. Each connection to a Natural host results in a new connection being made. Since each user requires a unique host session, connection pooling cannot be used. Therefore, you should make sure there are enough sessions for your users. The default maximum number is "32".
 6. Choose **Next**.
 7. Choose **Next**.
 8. Choose **Finish**.
 9. Open the tree node **Resources > Connectors > Connector Resources**.
 10. Choose **New**.
 11. Enter "eis/NaturalUnicodeRA" as the JNDI name.
 12. Select **NatPool** (or whatever name you specified previously) as the pool name.
 13. Choose **OK**.
- 6 Deploy the enterprise application *njx12.ear*:
 1. Open the tree node **Applications > Enterprise Applications**.

2. Choose **Deploy**.
 3. Select *njx12.ear* as the file to upload.
 4. Choose **Next**.
 5. Choose **OK**. The deployment may take several minutes.
- 7 Copy the license file into the directory `<sunas>/domains/domain1/applications/j2ee-apps/njx12/cisnatural_war/cis/licensekey` (you have to create the directory if it does not yet exist).
 - 8 Restart the application server.

Update Installation

► To update Natural for Ajax

- 1 Shut down the application server.
- 2 Create a backup copy of your *sessions.xml* file, which is located in `<sunas>/domains/domain1/applications/j2ee-apps/njx<nnn>/cisnatural_war/WEB-INF`.
- 3 Create a backup copy of your license file, which is located in `<sunas>/domains/domain1/applications/j2ee-apps/njx<nnn>/cisnatural_war/cis/licensekey`.
- 4 Create backup copies of all modified configuration files, which are located in `<sunas>/domains/domain1/applications/j2ee-apps/njx<nnn>/cisnatural_war/cis/config`.
- 5 Create backup copies of previously created projects, which are located in `<sunas>/domains/domain1/applications/j2ee-apps/njx<nnn>/cisnatural_war`.
- 6 Start the application server.
- 7 Start a web browser and enter the following URL:

```
http://<host>:<adminport>
```

This opens the Administration Console.

- 8 Undeploy the resource adapter *njx<nnn>ra.rar*.
- 9 Undeploy the enterprise application *njx<nnn>.ear*.
- 10 Deploy the new version of Natural for Ajax as in a first-time installation.
- 11 Shut down the application server.
- 12 Restore the files that you have backed up in steps 2, 3, 4 and 5.
- 13 Start the application server.
- 14 Start a web browser and enter the following URL:

```
http://<host>:<port>/cisnatural
```

This opens the Application Designer development workplace.

- 15 In the **Development Tools** node of the navigation frame, choose **Layout Manager**.
- 16 For each application project that you have created with an earlier release of Natural for Ajax, select the layout definitions and from the **Operations on multiple Items** menu, choose **(Re)Generate HTML Pages**.

Verifying the Installation

It is assumed that `http://<host>:<port>` is the URL of your application server.

To verify the installation

- 1 Enter the following URL in your web browser:

```
http://<host>:<port>/cisnatural
```

This opens the Application Designer development workplace.

- 2 Enter the following URL in your web browser:

```
http://<host>:<port>/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

This opens the Natural logon page. The installation is now complete.

4

Setting Up Your Environment

■ Setting Up Application Designer	20
■ Setting Up Your Development Environment for Natural	20
■ Setting Up Your Runtime Environment for Natural	21

Before you start developing and executing Natural for Ajax applications, you have to make specific definitions in your environment.

Setting Up Application Designer

Currently, there is nothing to configure for Natural pages.

Setting Up Your Development Environment for Natural

If you are practising remote development with Natural's Single Point of Development (SPoD), a Natural Development Server must be installed and activated on the remote machine.

■ Mainframe

When your Natural Development Server is located on a mainframe, see the Natural Development Server documentation.

■ UNIX

When your Natural Development Server is located on UNIX, see *Activating the Natural Development Server on UNIX* in the *Installation* documentation which is provided with Natural for UNIX.

■ Windows

When your Natural Development Server is located on Windows, the **Web I/O Interface service** option, which can be set with the setup type **Custom**, must be selected when installing Natural. See the *Installation* documentation which is provided with Natural for Windows.

▶ To set up Natural Studio

- 1 Ask your administrator for the host name and the port number of the Natural Development Server.
- 2 Connect to the Natural Development Server. See *Accessing a Remote Development Environment* in the *Remote Development Using SPoD* documentation which is provided with Natural for Windows.
- 3 It is recommended that you create a new Natural library for each Application Designer project.

▶ To set up Natural for Eclipse

- 1 Ask your administrator for the host name and the port number of the Natural Development Server.
- 2 Create a new target in Natural for Eclipse, using this host name and port number. For further information, see the Natural for Eclipse documentation.

- 3 When creating a Natural project, assign this target in the project properties.

Setting Up Your Runtime Environment for Natural

The following must be installed on the remote machine where you are going to test and execute the Natural code:

■ Mainframe

When your Natural Development Server is located on a mainframe, the Natural Web I/O Interface server must be installed and started. For detailed information, see *Installing and Configuring the Natural Web I/O Interface Server* in the *Natural Web I/O Interface* documentation which is provided for Natural for Mainframes.

■ UNIX

On UNIX, the Natural Web I/O Interface server is implemented as a daemon.

When your Natural Development Server is located on UNIX, the Natural Web I/O Interface daemon must be installed and activated. For detailed information, see *Installing and Configuring the Natural Web I/O Interface Server* in the *Natural Web I/O Interface* documentation which is provided for Natural for UNIX.

■ Windows

On Windows, the Natural Web I/O Interface server is implemented as a service.

When your Natural Development Server is located on Windows, the **Web I/O Interface service** option, which can be set with the setup type **Custom**, must be selected when installing Natural Runtime. See the *Installation* documentation which is provided with Natural for Windows.

See also *Installing and Configuring the Natural Web I/O Interface Server* in the *Natural Web I/O Interface* documentation which is provided for Natural for Windows.

▶ To set up the runtime environment for Natural for Mainframes

- 1 Ask your administrator for the host name and the port number of the Natural Web I/O Interface server.
- 2 Invoke the configuration tool which is used for managing the session configurations in the file *sessions.xml*. See *Using the Configuration Tool* in the *Configuring the Client* part of the *Natural Web I/O Interface* documentation.

- 3 Add a new session with the following settings:

Option	Description
Session ID	Enter the name that is to be available for selection in the logon page.
Host name	The host name of the Natural Web I/O Interface server. Enter the value that you have received from your administrator.
Port number	The port number of the Natural Web I/O Interface server. Enter the value that you have received from your administrator.

- 4 In the configuration file, there is a preconfigured session with the name "Natural for Ajax Examples". It contains dummy settings for the host name, port number and application. This session is intended to start the Natural for Ajax examples that are delivered with Natural in the library `SYSEXNJX`.

Enter the settings (host name and port number) that match your environment. Remove the dummy setting for the application (which is "script-name").

Then you will be able to execute the examples from the logon page.

- 5 Restart the application server.

► To set up the runtime environment for Natural for UNIX

- 1 Ask your administrator for the host name and the port number of the Natural Web I/O Interface server and the name of the script that is used to start up Natural sessions. A sample shell script for starting up Natural (*nwo.sh*) is delivered with Natural for UNIX; see also *nwo.sh - Shell Script for Starting Natural* in the *Natural Web I/O Interface* documentation.
- 2 Invoke the configuration tool which is used for managing the session configurations in the file *sessions.xml*. See *Using the Configuration Tool* in the *Configuring the Client* part of the *Natural Web I/O Interface* documentation.
- 3 Add a new session with the following settings:

Option	Description
Session ID	Enter the name that is to be available for selection in the logon page.
Host name	The host name of the Natural Web I/O Interface server. Enter the value that you have received from your administrator.
Port number	The port number of the Natural Web I/O Interface server. Enter the value that you have received from your administrator.
Application	The name of the script that is used to start up Natural sessions. Enter the value that you have received from your administrator.

- 4 In the configuration file, there is a preconfigured session with the name "Natural for Ajax Examples". It contains dummy settings for the host name, port number and application. This

session is intended to start the Natural for Ajax examples that are delivered with Natural in the library SYSEXNJX.

Edit this session and enter the settings (host name, port number and the name of the Natural startup script) that match your environment. Then you will be able to execute the examples from the logon page.

- 5 Restart the application server.

► To set up the runtime environment for Natural for Windows

- 1 Ask your administrator for the host name and the port number of the Natural Web I/O Interface server and the name of the batch file that is used to start up Natural sessions. A sample batch file for starting up Natural (*nwo.bat*) is delivered with Natural for Windows; see also *Batch File for Starting Natural* in the *Natural Web I/O Interface* documentation.
- 2 Invoke the configuration tool which is used for managing the session configurations in the file *sessions.xml*. See *Using the Configuration Tool* in the *Configuring the Client* part of the *Natural Web I/O Interface* documentation.
- 3 Add a new session with the following settings:

Option	Description
Session ID	Enter the name that is to be available for selection in the logon page.
Host name	The host name of the Natural Web I/O Interface server. Enter the value that you have received from your administrator.
Port number	The port number of the Natural Web I/O Interface server. Enter the value that you have received from your administrator.
Application	The name of the batch file that is used to start up Natural sessions. Enter the value that you have received from your administrator.

- 4 In the configuration file, there is a preconfigured session with the name "Natural for Ajax Examples". It contains dummy settings for the host name, port number and application. This session is intended to start the Natural for Ajax examples that are delivered with Natural in the library SYSEXNJX.

Enter the settings (host name, port number and the name of the Natural startup batch file) that match your environment. Then you will be able to execute the examples from the logon page.

- 5 Restart the application server.

5

First Steps

This part is organized under the following headings:

- **About this Tutorial**
- **Starting the Development Workplace**
- **Creating a Project**
- **Getting Started with the Layout Painter**
- **Writing the GUI Layout**
- **Setting Up Your Development and Runtime Environment for Natural**
- **Creating the Natural Code**
- **Some Background Information**

It is important that you work through the exercises in the same sequence as they appear in this tutorial. Problems may occur if you skip an exercise.

6 About this Tutorial

This tutorial provides an introduction to working with Natural for Ajax. It explains how to create a “Hello World!” application. This covers all basic steps you have to perform when creating pages with Natural for Ajax: you create a layout file, you create an adapter and a main program, and you run the application.

When you have completed all steps of this tutorial, the page for your “Hello World!” application will look as follows:



Preview

Hello World!

Say Hello

Hello

Input Area

Your Name

Output Area

Result

Input your name and press the 'Say Hello' button.

Your application will act in the following way: When you enter a name in the **Your Name** field and choose the **Say Hello** button, the **Result** field displays "Hello World" and the name you have entered.

To reach this goal, you will proceed as follows:

1. You will first create a new Application Designer project.
2. You will then use Application Designer's Layout Painter to create the following layout:



This corresponds to the following XML layout:

```

<?xml version="1.0" encoding="UTF-8"?>
<natpage natsource="HELLO-A">
  <titlebar name="Hello World!">
  </titlebar>
  <header withdistance="false">
    <button name="Say Hello" method="sayHello">
    </button>
  </header>
  <pagebody>
    <itr takefullwidth="true">
      <hdist width="100%">
      </hdist>
      <icon image="../../../cis demos/images/hello.gif">
      </icon>
    </itr>
    <rowarea name="Input Area">
      <itr>
        <label name="Your name" width="100">
        </label>
        <field valueprop="name" width="200">
        </field>
      </itr>
    </rowarea>
    <rowarea name="Output Area">
      <itr>
  
```

```
        <label name="Result" width="100">
        </label>
        <field valueprop="result" width="200" displayonly="true">
        </field>
    </itr>
</rowarea>
<vdist pixelheight="100">
</vdist>
<itr>
    <label name="Input your name and press the &apos;Say Hello&apos;
button." asplaintext="true">
    </label>
</itr>
</pagebody>
<statusbar withdistance="false">
</statusbar>
</natpage>
```

3. When you save your layout for the first time, an intelligent HTML page and the Natural adapter for this page are generated.
4. Before you can start coding, you have to make specific definitions in your development environment (this tutorial assumes that you are using Natural Studio as your development environment).
5. You will import the generated Natural adapter into your Natural library.
6. You will then create the main program which will use the adapter to display the page and which will handle the events that occur on the page, for example, when you choose the **Say Hello** button of your application.

You can now proceed with your first exercise: [Starting the Development Workplace](#).

7 Starting the Development Workplace

This tutorial assumes that you have installed Natural for Ajax as described in the *Installation* section.

▶ To start the development workplace

- 1 Make sure that your application server is running.
- 2 Invoke your browser and start the development workplace with the following URL:

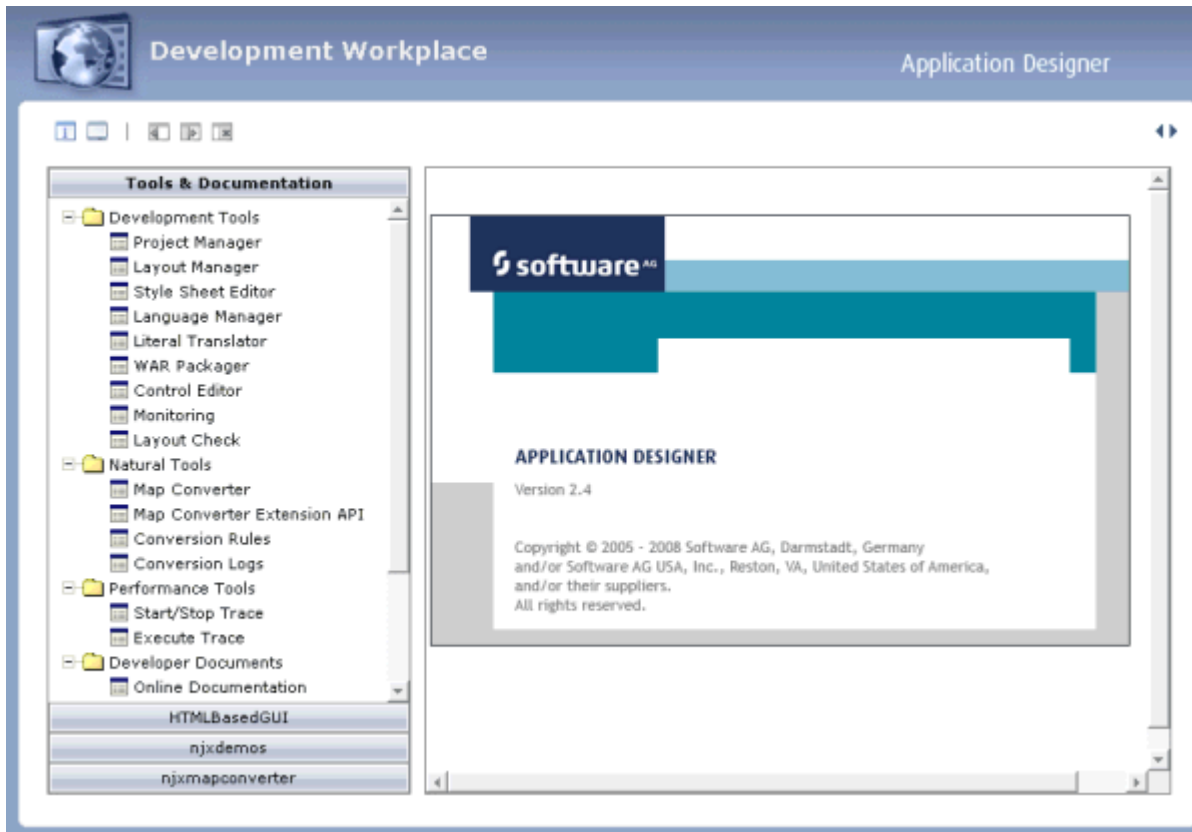
```
http://<host>:<port>/cisenatural
```

where *<host>* is the name of the machine on which your application server is installed and *<port>* is the port number of your application server.



Note: If you have not defined another port number during installation, the default port number is "8080".

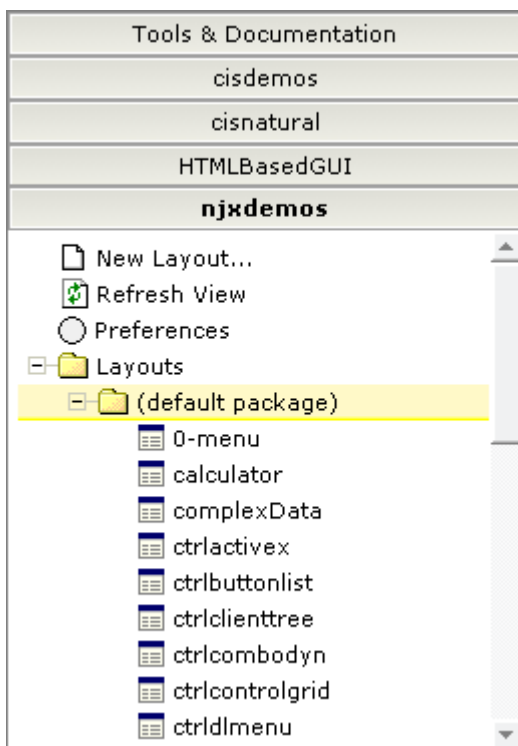
The development workplace is now shown in your browser.



You can now proceed with the next exercise: *Creating a Project*.

8 Creating a Project

In the Application Designer environment, layouts are structured in so-called application projects. In the development workplace, you see the existing projects on the left. For each project, there is a tree of layout definitions that you can display when you choose the button containing the project name. For example:



For this tutorial, you will now create a project with the name "cisnatfirst".


► To create a project

- 1 Choose **Tools & Documentation** to display the list of development tools.
- 2 Choose **Project Manager** in the tree.

A list of existing application projects is now shown on the right.

- 3 Choose the **New** button which is located below the list of application projects.

The following is now shown:



The screenshot shows a dialog box titled "Create New Application Project". It has a text input field labeled "Application Project" and a "Create" button. Below the button, there is a note: "Please note: you have to create a context root inside your servlet engine! The name of the context root is the name of the application project. In the native Application Designer environment this is done automatically by restarting the Server."

- 4 Enter "cisnatfirst" as the name of your project and choose the **Create** button.

Your new project is now shown in the list of existing application projects on the right.

The left side, which shows buttons for all existing projects, now also shows a button for your new project.

You can now proceed with the next exercise: [Getting Started with the Layout Painter](#).

9

Getting Started with the Layout Painter

■ Creating a New Layout	36
■ Elements of the Layout Painter Screen	38
■ Previewing the Layout	39
■ Viewing the XML Code	40

The Layout Painter, which can be accessed from the development workplace, is used to write the page layout. This is an Application Designer application itself.

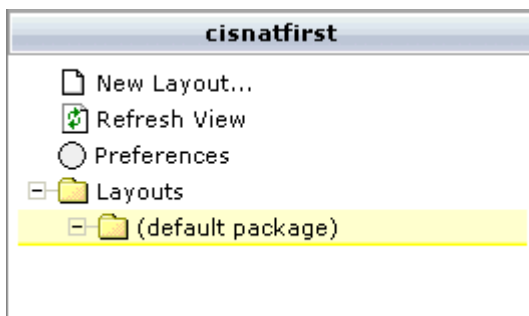
Creating a New Layout

You will now create a layout which is stored in the project you have previously created.

► To choose a layout template

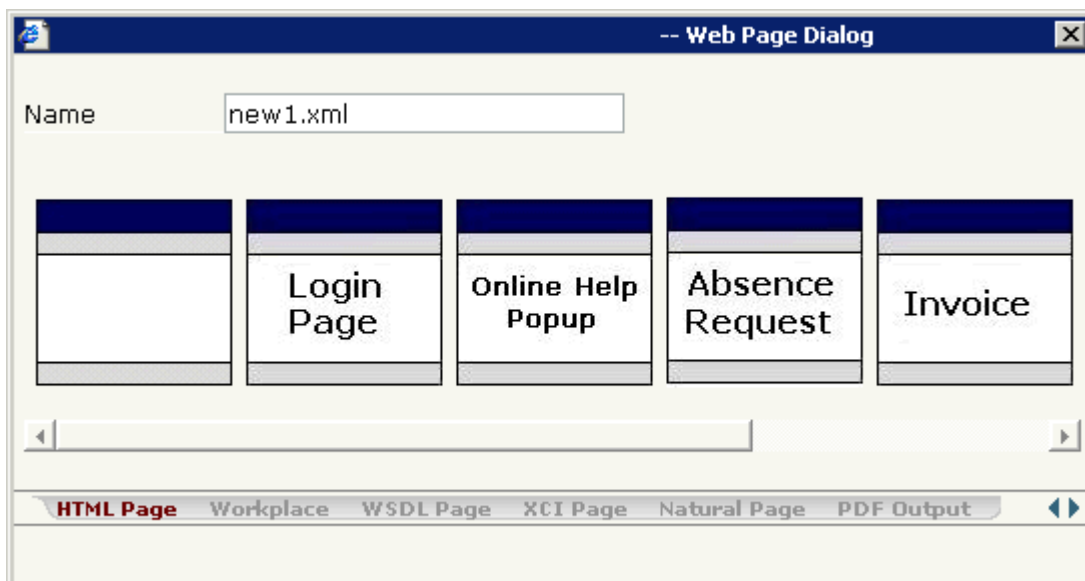
- 1 Choose the button for the project **cisnatfirst**.

The list of layout nodes inside the tree will be empty at the beginning:



- 2 Choose **New Layout...** in the tree.

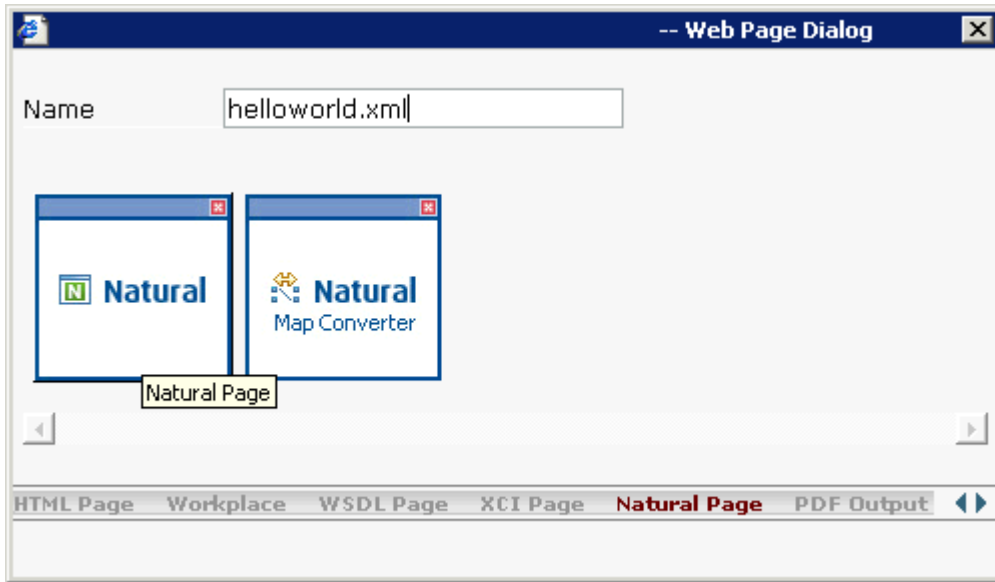
The following dialog appears.



- 3 Enter "helloworld.xml" in the **Name** text box.

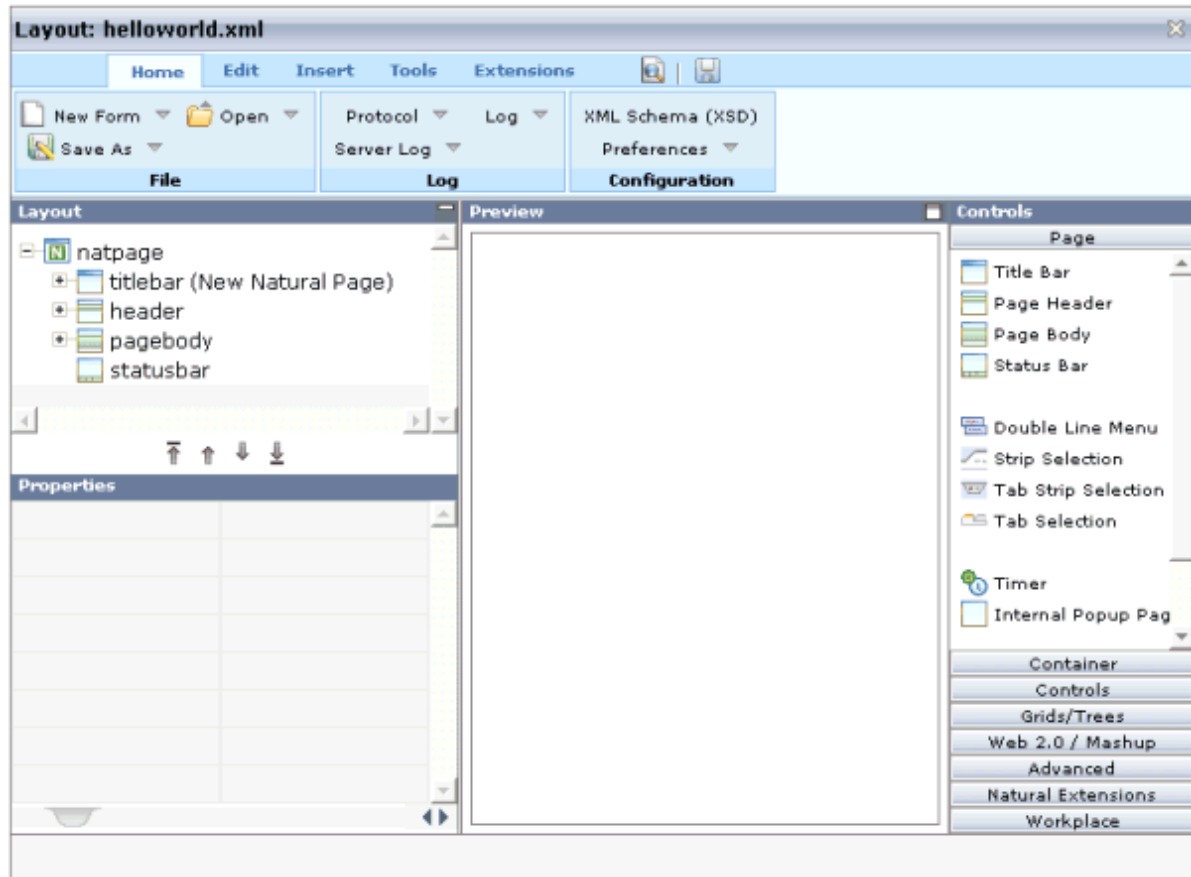
This is the name of your layout definition.

- 4 Select the **Natural Page** tab at the bottom of the dialog.



- 5 Select the template for the Natural page (when you move the mouse over this template, the tool tip "Natural Page" appears).

The main screen of the Layout Painter appears:



Note: The file *helloworld.xml* is stored in the */xml* directory of your project.

Elements of the Layout Painter Screen

The Layout Painter screen is divided into several areas:

■ Layout Area (left side)

This area consists of a layout tree and a properties area.

The layout tree contains the controls that represent the XML layout definition. You drag these controls from the controls palette into the layout tree. Each node in the layout tree represents an XML tag.

In the properties area below the layout tree, you specify the properties for the control which is currently selected in the layout tree.

- **Preview Area (middle)**

The preview area shows the HTML page which is created using the controls in the layout area. This page is refreshed each time, you choose the preview button (see below).

- **Controls Palette (right side)**

Each control is represented by an icon. A tool tip is also provided which appears when you move the mouse pointer over the control. This tool tip also displays the XML tag which will be used in the XML layout.

The palette is structured into sections, where each section represents a certain type of controls.

Previewing the Layout

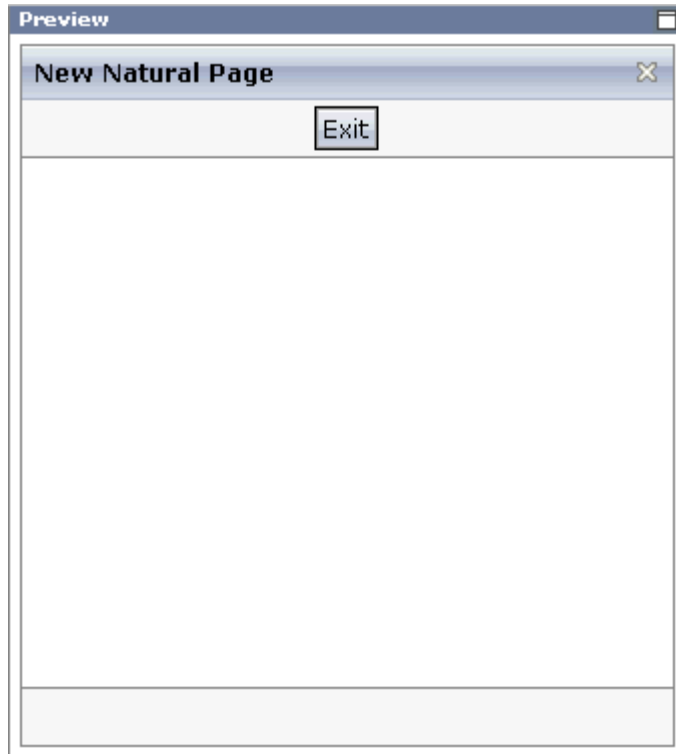
The layout tree inside the Layout Painter already contains some nodes that were copied from the template that you chose in the dialog in which you specified the name of the page. To see what the page looks like, preview the layout as described below.

- ▶ **To preview the layout**

- Choose the following button which is shown at the top of the Layout Painter.



The preview area is updated and you see the page. The page already contains a title bar, a header containing an **Exit** button, the page body and a status bar.



The preview area is a sensitive area. When you select a control in the preview area (for example, the title bar), this control is automatically selected in the layout tree.

Viewing the XML Code

When creating the layout, you can view the currently defined XML code.

► To view the XML code

- From the **Edit** tab of the Layout Painter, choose **XML**.

A dialog box appears. At this stage of the tutorial, it contains the following XML layout definition for the nodes which were copied from the template.

```
<natpage natsinglebyte="true"
xmlns:njx="http://www.softwareag.com/njx/njxMapConverter">
  <titlebar name="New Natural Page">
  </titlebar>
  <header withdistance="false">
    <button name="Exit" method="onExit">
    </button>
  </header>
</pagebody>
```

```
</pagebody>  
<statusbar withdistance="false">  
</statusbar>  
</natpage>
```

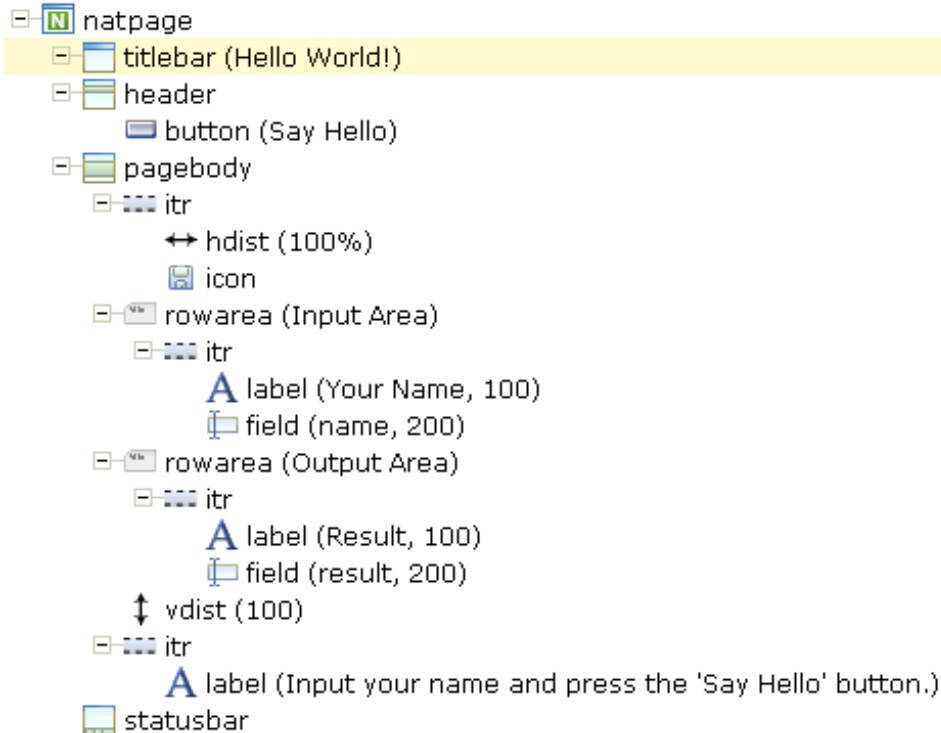
You can now proceed with the next exercise: *Writing the GUI Layout*.

10

Writing the GUI Layout

■ Specifying the Properties for the Natural Page	44
■ Specifying a Name for the Title Bar	45
■ Using the Property Editor	46
■ Specifying a Name and Method for the Button	48
■ Adding the Input and Output Areas	48
■ Adding the Image	52
■ Adding a Horizontal Distance	52
■ Adding an Instructional Text	53
■ Adding a Vertical Distance	54
■ Saving Your Layout	54

You will now create the layout for your “Hello World!” application. When you have completed all exercises in this chapter, the layout should look as shown below and the **XML code** should be the same as shown in the section *About this Tutorial*.



Tip: **Preview the layout** and **view the XML code** each time you have completed an exercise. If the system finds some wrong or missing definitions while generating the preview page, there will be a corresponding message in the status bar. From the **Home** tab of the Layout Painter, choose **Protocol** to get more information about these problems.

Specifying the Properties for the Natural Page

You will now specify the following for the Natural page:

■ Name for the Natural Adapter (natsource)

The value in the property `natsource` defines the name of the adapter. The adapter is a Natural object that your application will use to communicate with the page. It will be generated when you save the page layout.

If you do not specify a value for `natsource`, the name that you have specified for the layout (without the extension ".xml") will be used as the name for the Natural adapter. If you want to use the adapter in a development environment other than Natural for Eclipse, you must make sure that the resulting name matches the naming conventions for Natural object names.

■ Handling of Strings (`natsinglebyte`)

Using the property `natsinglebyte`, you can specify how the strings displayed on this page are to be handled in the Natural application. Natural knows two types of strings: Unicode strings (format U) and code page strings (format A). By default, the strings displayed in web pages are mapped to Unicode strings in Natural. For this tutorial, you will specify that code page strings are to be used. Therefore, you will set the property `natsinglebyte` to "true".

If you do not specify a value for `natsinglebyte` or when you set it to "false", Unicode strings will be used.

► To specify the properties for the Natural page

- 1 In the layout tree, select the node **natpage**.

The properties for this control are now shown in the properties area at the bottom.

- 2 Specify the following properties:

Property	Value
<code>natsource</code>	HELLO-A
<code>natsinglebyte</code>	true

Specifying a Name for the Title Bar

You will now specify the string "Hello World!" which is to appear in the title bar of your application.

► To specify the name for the title bar

- 1 In the layout tree, select the node **titlebar (New Natural Page)**.

The properties for this control are now shown in the properties area at the bottom. You can see the default entry "New Natural Page" for the `name` property.

- 2 Specify the following property:

Property	Value
<code>name</code>	Hello World!

When you click on the layout tree, the node in the layout tree changes to **titlebar (Hello World!)**.



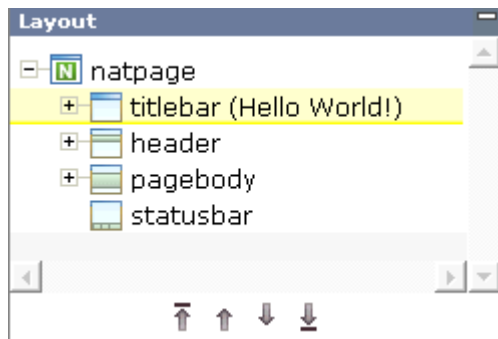
Note: Properties that are left blank are not shown in the XML code.

Using the Property Editor

You can also specify the property values using the Property Editor. In this case, you can access detailed help information on each property.

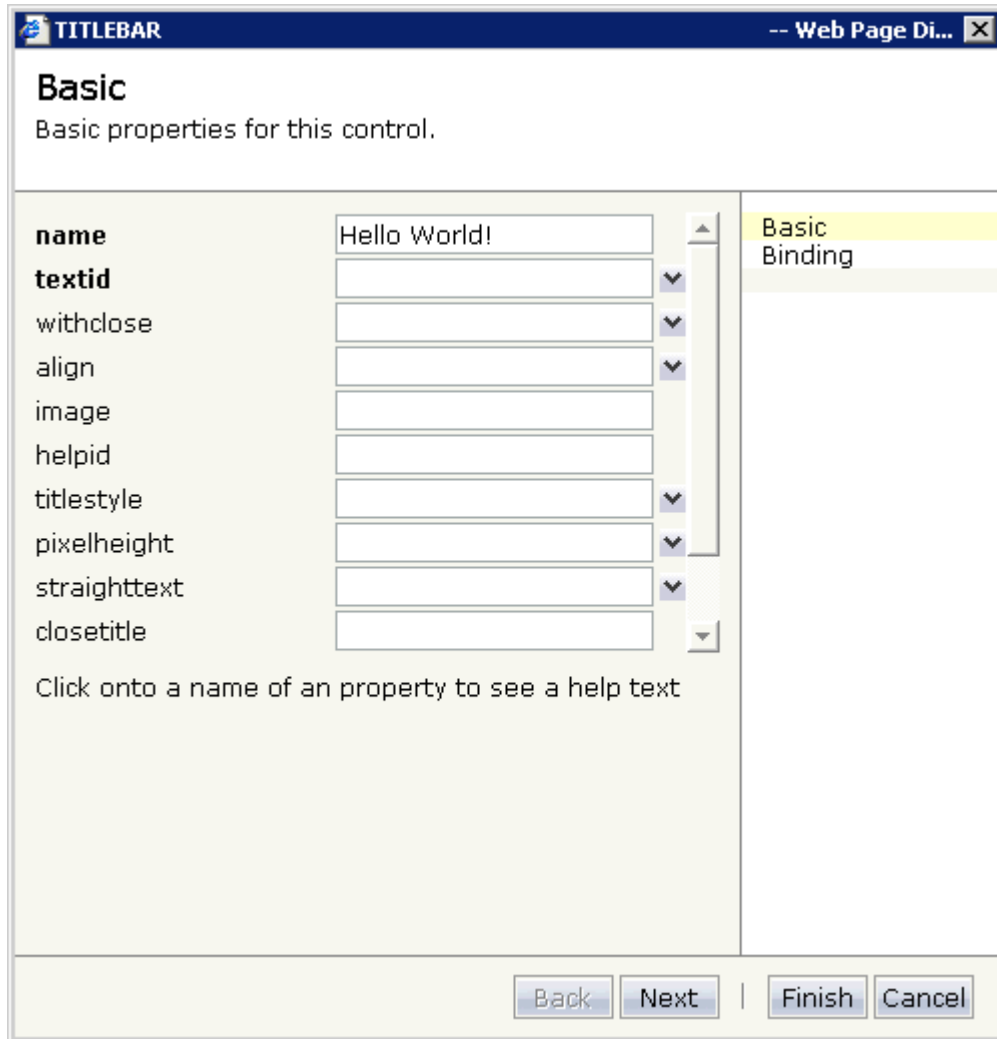
► **To use the Property Editor**

- 1 Select the control in the layout tree for which you need help, for example, the **titlebar (Hello World!)** node.



- 2 From the **Edit** tab of the Layout Painter, choose **Property Editor**.

The following dialog appears.



The properties of the control are listed.

- 3 Click on the name of a property to display detailed information on this property. This information is shown below the list of properties.
- 4 Choose the **Finish** button to close the dialog.

Any changes you have applied in the dialog will be saved.

Specifying a Name and Method for the Button

You will now specify the string "Say Hello" which is to appear on the button. And you will specify the name of the method that is to be invoked when the user chooses this button.

► To specify the name and the method for the button

- 1 In the layout tree, open the **header** node.



Note: By clicking the icon of a node, you hide or expand the node's subnodes.

You can now see the entry for the button with the default name "Exit".

- 2 Select the node **button (Exit)**.
- 3 Specify the following properties:

Property	Value
name	Say Hello
method	sayHello

The method needs to be programmed in the adapter. This will be explained later in this tutorial.

Adding the Input and Output Areas

The input and output areas in this tutorial are created using **Row Area** controls. These controls can be found in the **Container** section of the controls palette.

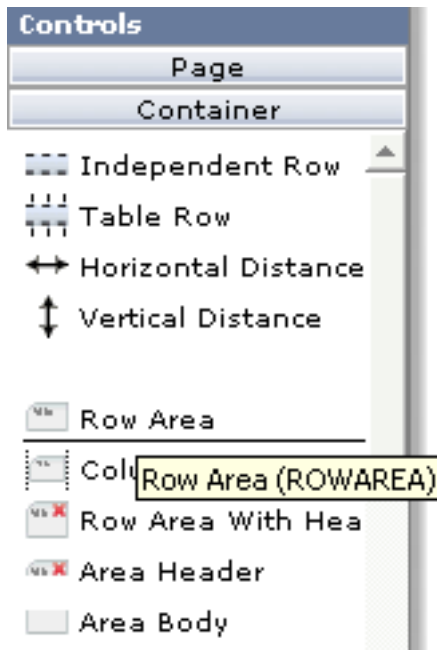
Each row area will contain an **Independent Row** control which in turn contains a **Label** and a **Field** control. These controls can be found in the **Controls** section of the controls palette.

For adding controls to your layout, you drag them from the controls palette onto the corresponding tree node in the layout tree. This is explained below.

► To create the input area

- 1 Open the **Container** section of the controls palette.

When you move the mouse over a control, a tool tip appears which also displays the control name which will be used in the XML layout. For example:



- 2 Drag the **Row Area** control from the controls palette onto the **pagebody** node in the layout tree.

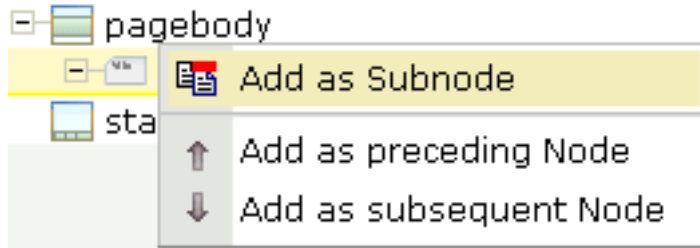
The row area is added as a subnode of the **pagebody** node. The new subnode is automatically selected so that you can maintain the properties of the row area directly in the properties area.

- 3 Specify the following property:

Property	Value
name	Input Area

- 4 Drag the **Independent Row** control from the controls palette onto the **rowarea (Input Area)** node in the layout tree.

When you drop information into the tree, the system will sometimes respond by offering a context menu with certain options about where to place the control. In this case, the following context menu appears.



Note: When you move the mouse outside the context menu, the context menu disappears. The control is not inserted in this case.

- 5 Choose the **Add as Subnode** command.

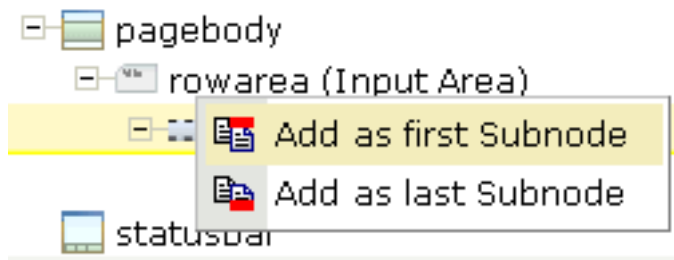
The control is now inserted below the **rowarea (Input Area)** node. The new node is shown as **itr**.

- 6 Open the **Controls** section of the controls palette.
- 7 Drag the **Label** control from the controls palette onto the **itr** node you have just inserted and specify the following properties:

Property	Value
name	Your Name
width	100

- 8 Drag the **Field** control from the controls palette onto the **itr** node you have just inserted.

A context menu appears and you have to specify where to place the control.



- 9 From the context menu, choose the **Add as last Subnode** command.

10 Specify the following properties for the field:

Property	Value
valueprop	name
width	200

► To create the output area

- Create the output area in the same way as the input area (add it as the last subnode of the **pagebody** node), with the following exceptions:

Row Area

Specify a different value for the following property:

Property	Value
name	Output Area

Label

Specify a different value for the following property:

Property	Value
name	Result

Field

Specify different values for the following properties:

Property	Value
valueprop	result
displayonly	true



Note: To display the `displayonly` property, choose the **Appearance** tab at the bottom of the properties area. You can then select the required value from a drop-down list box.

Adding the Image

You will now add the image which is to be shown above the input area. To do so, you will use the **Icon** control which can be found in the **Controls** section of the controls palette.



Note: The image is provided in Application Designer's */cisdemos/images* directory.

▶ To add the image

- 1 Drag the **Icon** control from the controls palette onto the **pagebody** node in the layout tree.

The icon is added as the last subnode of the **pagebody** node. It is automatically placed into an **itr** (independent row) node.

- 2 Specify the following property for the icon:

Property	Value
image	../cisdemos/images/hello.gif

- 3 Select the **itr** node containing the icon and choose the following button below the layout tree:



The selected node is now moved up so that it appears as the first subnode of the **pagebody** node.

- 4 Specify the following property for the **itr** node:

Property	Value
takefullwidth	true

Adding a Horizontal Distance

When you preview the layout, you will see that the image you have just added appears centered.

You will now move the image to the right side of the page. To do so, you will use the **Horizontal Distance** control which can be found in both the **Controls** section and the **Container** section of the controls palette.

► To add the horizontal distance

- 1 Drag the **Horizontal Distance** control from the controls palette onto the **itr** node containing the icon.
- 2 From the resulting context menu, choose the **Add as first Subnode** command.

The node **hdist** is inserted into the tree.

- 3 Specify the following property:

Property	Value
width	100%

Adding an Instructional Text

You will now enter a text which is to appear below the output area and which tells the user what to do.

To do so, you will once again use the **Independent Row** control into which you will insert a **Label** control.



Note: The **Independent Row** control can be found in both the **Controls** section and the **Container** section of the controls palette.

► To add the independent row with the label

- 1 Drag the **Independent Row** control from the controls palette onto the **pagebody** node in the layout tree.
- 2 From the resulting context menu, choose the **Add as last Subnode** command.

The node **itr** is inserted into the tree.

- 3 Drag the **Label** control from the controls palette onto the **itr** node you have just created.

- 4 Specify the following properties for the label:

Property	Value
name	Input your name and press the 'Say Hello' button.
asplaintext	true



Note: Go to the **Appearance** tab to display the property `asplaintext`.

Adding a Vertical Distance

When you preview the layout, you will see that the text you have just added appears directly below the output area. You will now move the text 100 pixels to the bottom.

To do so, you will use the **Vertical Distance** control which can be found in both the **Controls** section and the **Container** section of the controls palette.

▶ To add the vertical distance

- 1 Drag the **Vertical Distance** control from the controls palette onto the **itr** node containing the label.
- 2 From the resulting context menu, choose the **Add as preceding Node** command.

The node **vdist** is inserted into the tree.

- 3 Specify the following property:

Properties	Value
height	100

Saving Your Layout

If you have not already done so, you should now save your layout.

When you save a layout for the first time, an HTML file is generated (in addition to the XML file) which is placed into the root directory of your application project. This HTML file is updated each time you save the layout.

The Natural adapter is also created when you save your layout for the first time. Later in this tutorial, you will import this adapter into your Natural library. Your application program will use the adapter to communicate with the page.

► **To save the layout**

- Choose the following button which is shown at the top of the Layout Painter.



You can now proceed with the next exercise: *[Setting Up Your Development and Runtime Environment for Natural](#)*.

11

Setting Up Your Development and Runtime Environment for Natural

Before you start coding, you have to make specific definitions in your Natural environment.

► To set up your Natural environment

- Set up your Natural development and runtime environment for the required platform as described in *Setting Up Your Environment* previously in this documentation.

This tutorial assumes that you use Natural Studio as your development environment.

Make sure to use the names mentioned below.

■ Development Environment

Create a new Natural library with the name `CISHELLO`.

■ Runtime Environment

When you add the new entry to the configuration file, specify "Execute samples" as the session name:

```
<session id="Execute samples" trace="false">
```

"Execute samples" is the entry that will later be available for selection in the logon page.

You can now proceed with the next exercise: *Creating the Natural Code*.

12

Creating the Natural Code

■ Importing the Adapter into Natural	60
■ Creating the Main Program	61
■ Testing the Completed Application	64

Importing the Adapter into Natural

You will now import the generated adapter into Natural to make it available to your application.

When you saved your page layout, Application Designer created the Natural adapter HELLO-A for your page. This is the name that you have specified earlier in this tutorial. Your application program will use the adapter to communicate with the page. The adapter has been generated into the following directory:

`<installdir>/cisnatfirst/nat`



Note: The location of `<installdir>` depends on your application server environment.

▶ To import the adapter

- 1 Import the adapter source into the Natural library CISHELLO which you have created earlier in this tutorial. To do so, use either drag-and-drop or the import function of the SYSMAIN utility.

The adapter looks as follows:

```
* PAGE1: PROTOTYPE      --- CREATED BY Application Designer --- /*<R0>>
* PROCESS PAGE USING 'XXXXXXXX' WITH
* NAME RESULT
* DEFINE DATA PARAMETER
* 1 NAME (U) DYNAMIC
* 1 RESULT (U) DYNAMIC
* END-DEFINE
*
* PROCESS PAGE U'/cisnatfirst/helloworld' WITH
* PARAMETERS
*   NAME U'name'
*   VALUE NAME
*   NAME U'result'
*   VALUE RESULT
* END-PARAMETERS
*
* TODO: Copy to your calling program and implement.
/*/*( DEFINE EVENT HANDLER
* DECIDE ON FIRST *PAGE-EVENT
*   VALUE U'nat:page.end'
*   /* Page closed.
*   IGNORE
*   VALUE U'sayHello'
*   /* TODO: Implement event code.
*   PROCESS PAGE UPDATE FULL
```

```

*   NONE VALUE
*   /* Unhandled events.
*   PROCESS PAGE UPDATE
*   END-DECIDE
/*/*) END-HANDLER
*
END /*<<R0>

```

- 2 Stow the adapter.

Creating the Main Program

You will now create the main program which uses the adapter to display the page and which handles its events. The name of the program will be HELLO-P and you will store it in the library CISHELLO.

This description assumes that you are working with Natural Studio.

▶ To create the main program

- 1 Make sure that the library CISHELLO is selected.
- 2 From the **Object** menu, choose **New > Program**.
- 3 Enter a `DEFINE DATA` statement:

```

DEFINE DATA LOCAL
END-DEFINE

```

- 4 Import the adapter interface into the `DEFINE DATA` statement:
 1. Place the cursor in `END-DEFINE`.
 2. From the **Program** menu, choose **Import**.
 3. In the resulting dialog box, select the **Adapter** option button.
 4. Select the object HELLO-A.
 5. Select all importable data fields.
 6. Choose the **Import** button.

The result is your completed `DEFINE DATA` statement:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
```

- 5 Enter the `PROCESS PAGE` statement. The statement uses the page adapter to display the page in the web browser and to pass data to the controls on the page:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
```

- 6 Initialize the page data. In the page layout definition, the property `name` has been bound to the `FIELD` control with the label **Your Name**. For the property `name`, a parameter `NAME` has been generated into the parameter data area of the adapter. Thus, in order to preset the `FIELD` control, we will preset the variable `NAME` with the value "Application Designer".

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
```

- 7 Handle the events that can occur on the page. A template for the event handler code has been generated as a comment block into the page adapter `HELLO-A`. List the adapter `HELLO-A` and copy this comment block into your main program and terminate the program with an `END` statement:

```
DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
*
DECIDE ON FIRST *PAGE-EVENT
VALUE 'nat:page.end'
```

```

/* Page closed.
  IGNORE
  VALUE 'sayHello'
/* TODO: Implement event code.
  PROCESS PAGE UPDATE FULL
  NONE VALUE
/* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*
END

```

After the page has been displayed, the user raises events on the page by using the controls. The name of the raised event is then contained in the system variable `*PAGE-EVENT`. Depending on the event, the program modifies the page data, resends it to browser with a `PROCESS PAGE UPDATE FULL` statement and waits for the next event to occur.

The predefined event `nat:page.end` is raised when the user closes the page. The event `sayHello` is raised when the user chooses the **Say Hello** button. Previously in this tutorial, you have bound the event `sayHello` to this button while designing the page. The `NONE VALUE` block should always be defined as above. It contains the default handling of all events that are not handled explicitly.

- 8 When the event `sayHello` occurs, we want to display a greeting in the `FIELD` control with the label **Result**. Therefore, we modify the variable `RESULT` (which is bound to the corresponding `FIELD` control in the page layout) accordingly before we resend the page data.

```

DEFINE DATA LOCAL
1 NAME (A) DYNAMIC
1 RESULT (A) DYNAMIC
END-DEFINE
*
NAME := 'Application Designer'
PROCESS PAGE USING 'HELLO-A'
WITH NAME RESULT
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE 'nat:page.end'
  /* Page closed.
    IGNORE
  VALUE 'sayHello'
  /* TODO: Implement event code.
    COMPRESS 'Hello, ' NAME '!' TO RESULT
    PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
    PROCESS PAGE UPDATE
END-DECIDE
*
END

```

The main program is now complete.

If you have not yet saved the program, save or stow it now with the name "HELLO-P".

- 9 Catalog all modules in the library CISHELLO.

Testing the Completed Application

You will now run the application in your web browser and check whether it provides the desired result.

The generated HTML file *helloworld.html* (which is updated each time you save your layout) can be found within the root of your application project, that is in `<installdir>/cisnatfirst`.

This HTML page has some prerequisites concerning the browser workplace in which it is running. Therefore, it is per se not usable as a directly accessible page but needs to be embedded into a frame providing a defined set of functions.

It is necessary to logon to Natural before starting an application. Therefore, Natural applications are started using a logon page.

To test the application

- 1 Enter the following URL inside your browser:

```
http://localhost:8080/cisnatural/servlet/StartCISPage?PAGEURL=/cisnatural/NatLogon.html
```

The logon page should now appear.

The screenshot shows a web-based logon interface for 'Natural'. It is divided into two main sections: 'Connection details' and 'Change Password'. Below these is a 'Connect' button.

Connection details

- Session ID: A dropdown menu with 'Execute Samples' selected.
- Host name: A text input field.
- Port: A text input field.
- User name: A text input field.
- Password: A text input field.
- Natural application: A text input field.
- Natural parameter: A text input field.
- Language: A dropdown menu with 'English' selected.

Change Password

- New password: A text input field.
- Repeat new password: A text input field.

Connect

If the logon page is not displayed, check the following:

- URLs are case-sensitive. Double-check your input.
 - Check whether the file *NatLogon.html* is available in the directory *cisnatural*.
- 2 On the logon page, select the entry **Execute samples** from the **Session ID** drop-down list box. You have prepared this entry earlier in this tutorial when you have set up the runtime environment.
 - 3 Provide your user ID and password valid for the machine on which the Natural application will be running.
 - 4 In the **Natural application** text box, enter the following information, depending on your Natural platform:
 - **Natural for Mainframes**
Enter the name of the Natural program that is to be started. In our case, this is `HELLO-P`.
 - **Natural for UNIX**
Enter the name of the UNIX shell script that is used to start Natural. By default, this is `nwo.sh`.
 - **Natural for Windows**
Enter the name of the Windows command file (`.bat`) that is used to start Natural. By default, this is `nwo.bat`.
 - 5 In the **Natural parameters** text box, enter the following information, depending on your Natural platform:

■ **Natural for Mainframes**

Enter the dynamic Natural profile parameters that are necessary to start your application:

```
STACK=(LOGON CISHELLO)
```



Note: With Natural for Mainframes, is recommended to specify the Natural program that starts the application in the **Natural application** text box instead of passing it with the profile parameter `STACK`.

■ **Natural for UNIX and Natural for Windows**

Enter the Natural command line that is necessary to start your application:

```
STACK=(LOGON CISHELLO;HELLO-P)
```

- 6 Choose the **Connect** button.

Your application should be started now.

- 7 Enter your name and choose the **Say Hello** button.

The page should now successfully “talk” to your adapter.

The screenshot shows a web application interface with two main sections. The top section is titled 'Input Area' and contains a label 'Your Name' followed by a text input field containing the text 'Jo'. The bottom section is titled 'Output Area' and contains a label 'Result' followed by a text input field containing the text 'Hello World, Jo !!!!'. Both sections have a light blue header and a light yellow body.

You have now completed this tutorial. See the remaining section of these *First Steps* for [some background information](#).

13

Some Background Information

■ Name Binding between Controls and Adapter	68
■ Data Exchange at Runtime	68
■ Files and their Locations	69

Name Binding between Controls and Adapter

Which are the critical parts when building the “Hello World!” application?

- The NATPAGE control in the layout points to the name of the adapter object (property `natsource`).
- The FIELD control in the layout points to the property name of the adapter (property `valueprop`).
- The BUTTON control in the layout points to the event `sayHello()` of the adapter (property `method`).

There is a name binding between the layout definition and its corresponding adapter. This is the simple and effective approach of the Application Designer's development process: The adapter represents a logical abstraction of what the page displays. All layout definitions are kept in the page - all the logic is kept in the adapter. (Or better: behind the adapter. The adapter itself should only be a facade to the “real” application logic.)

Data Exchange at Runtime

What happens at runtime?

- When the user starts a Natural session from the logon page, the Natural program that the user specified in the command line is started.
- The Natural program executes a `PROCESS PAGE` statement, using an adapter.
- The `PROCESS PAGE` statement passes the name of the HTML page to be used and the initial page data to the browser.
- The browser displays the page. JavaScript code on the page distributes the initial data to the controls.
- The user provides some input, for example, enters the name. The content change is stored inside the page. The Natural program is not yet involved.
- The user does something which causes a flush of the changes (for example, the user chooses a button). Therefore, all registered data changes are packaged and are sent through the adapter to the Natural program, including the information which event has been raised.
- The Natural program receives the modified data.
- The system variable `*PAGE-EVENT` receives the name of the raised event.
- The event handler in the Natural program modifies the data and resends it to the page using a `PROCESS PAGE UPDATE` statement.
- And so forth.

With a standard HTTP connection, only the changed content of the screen is passed when operating on one page. The layout is kept stable in the browser. Consequently, there is no flickering of the page due to page reloading.

All steps described in the list above are done completely transparent to your adapter; i.e. you do not have to cope with session management, stream parsing, error management, building up HTML on the server, etc. You just have to provide an intelligent HTML page by defining it in the Layout Painter and an adapter object.

Files and their Locations

Have a look at the files created for your “Hello World!” application and take notice of the directory in which they are located.

All files are located in the directory `<installdir>/cisenatural/cisenatfirst`. The `<installdir>/cisenatural` directory is the directory of the web application instance. The `<installdir>/cisenatural/cisenatfirst` directory is the directory that has been created for your new project.

- The XML layout definition is kept in the `<installdir>/cisenatural/cisenatfirst/xml` directory.
- The generated HTML page is kept directly in the project directory. There are also some other files inside this directory that start with “ZZZZ”. These files are temporary files used when previewing pages inside the Layout Painter.
- The generated Natural adapters are kept in the directory `<installdir>/cisenatural/cisenatfirst/nat`.
- In the directory `<installdir>/cisenatural/cisenatfirst/accesspath`, “access restriction” files are generated. If you view these files inside a normal text editor (such as Notepad), you see that one file is maintained for each page; it holds the information about which properties are accessed by the page.

14

Developing the User Interface

■ Starting the Development Workplace	72
■ Creating an Application Designer Project	73
■ Creating a Natural Page	73
■ Specifying Properties for the Natural Page	74
■ Designing the Page	75
■ Binding Properties and Methods	76
■ Previewing the Layout	76
■ Viewing the Protocol	77
■ Saving the Layout	77
■ Generating the Adapter	77
■ Data Type Mapping	78

In the *First Steps* tutorial, you have developed a small rich internet program step by step. In this tutorial, you have already performed most of the steps required to develop a rich internet application.

The general procedure to develop a rich internet application with Natural for Ajax is as follows:

1. Use Application Designer to design the web pages that form the user interface of your application.
2. Generate a Natural adapter for each page (by saving the page). The adapter is a Natural object that forms the interface between the application code and the web page.
3. Use one of the Natural tools (Natural Studio or Natural for Eclipse) to write the Natural application programs that contain the business logic and use adapters to exchange data with the web pages.

In this chapter, the first two steps (design and adapter) are explained in more detail. Step 3 (business logic) is described in the section *Developing the Application Code* which also addresses advanced topics that are not covered in the tutorial.

For detailed information on how to use the Application Designer development workplace, see *Development Tools* in the Application Designer documentation. The latest version of the Application Designer documentation is available at http://documentation.softwareag.com/webmethods/cit_reroute.htm. The information which is provided below describes the most important differences which pertain to Natural for Ajax.

Starting the Development Workplace

The Application Designer development workplace is the central point for starting tools for layout development.

► To start the development workplace

- 1 Make sure that your application server is running.
- 2 Invoke your browser and start the development workplace with the following URL:

```
http://<host>:<port>/cisinatural
```

where *<host>* is the name of the machine on which your application server is installed and *<port>* is the port number of your application server.



Note: If you have not defined another port number during installation, the default port number is "8080".

Creating an Application Designer Project

First you create an Application Designer project using the Project Manager. The project contains the layouts of the web pages you design, the files that are generated from the layouts and are required to run your application and additional files that make your application multi language capable and supply help information. See also [Creating a Project](#) in the tutorial.



Note: Detailed information on the Project Manager is provided in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

All files in your Application Designer project are stored in one directory on the application server where Natural for Ajax is installed. The name of the directory corresponds to the project name you have chosen. The location of the directory depends on the application server:

- **JBoss Application Server**

`<installdir>/server/default/deploy/njx<nnn>.ear/cisnatural.war`

- **Sun Java System Application Server**

`<installdir>/domains/domain1/applications/j2ee-apps/njx<nnn>.ear/cisnatural_war`

where `<installdir>` is the directory in which your application server is installed and `<nnn>` is the current Natural for Ajax version.

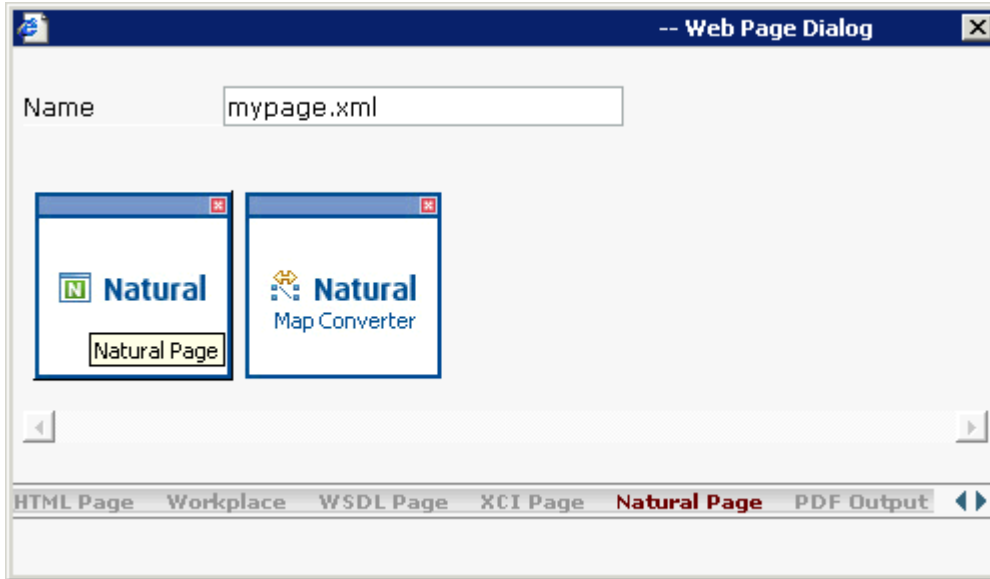
Creating a Natural Page

In order to create the layout of your web pages, you use Application Designer's Layout Painter.



Note: Detailed information on the Layout Painter is provided in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

Add a page layout to your project as described in [Creating a New Layout](#) in the tutorial (select the template for the Natural page).

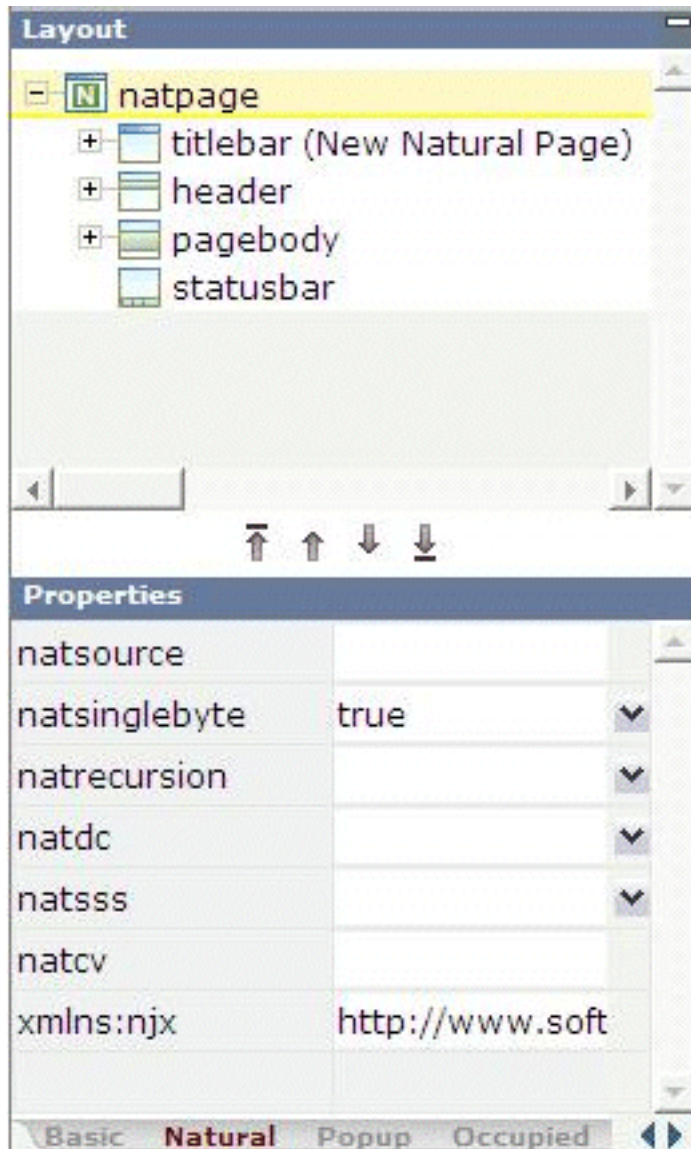


Note: More detailed information on creating a layout is provided in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

Specifying Properties for the Natural Page

In order to specify generation options for the new page, you specify values for certain properties that are specific for Natural pages.

To define properties, you select the node **natpage** in the layout tree of the Layout Painter. The properties for this control are then shown in the properties area at the bottom. When you select the **Natural** tab in the properties area, you can see the Natural-specific properties.



For information on the properties that are available for a Natural page, see [NATPAGE](#).

Designing the Page

Design your Natural page by dragging controls and containers from the controls palette onto the corresponding node in the layout tree or to the HTML preview. This has already been explained in the section [Writing the GUI Layout](#) of the tutorial.



Note: More detailed information on defining the layout is provided in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

Binding Properties and Methods

Many of the controls you use on your page have properties that can be controlled by the application. Also the controls can raise events that your application may wish to handle. The next step is therefore assigning identifiers to each of these properties and events under which your application can later address them. This procedure is called “binding”.

To get an overview which properties and events are bindable to application variables and events, it is a good idea to select a control in the layout tree and open the Event Editor as described in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

The Event Editor displays only those properties of controls that can be bound to application variables and events. It indicates also which properties must be bound mandatorily. The usage and meaning of each of the properties and events is described for each control in the following sections of this Natural for Ajax documentation:

- *Working with Controls*
- *Working with Grids*
- *Working with Trees*
- *Working with Menus*
- *Non-Visual Controls and Hot Keys*

As an example for property and event binding, see the following sections in the *First Steps* tutorial:

- *Using the Property Editor*
- *Specifying a Name and Method for the Button*

Previewing the Layout

To find out how the current layout definitions are rendered on the page, preview the layout as described in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

Viewing the Protocol

The protocol contains warnings and error messages that might occur while you design and preview your page. For further information, see the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

Saving the Layout

Save the page layout as described in [Saving Your Layout](#) in the tutorial.

Other than with Java adapters (which are described in the Application Designer documentation), you do not use the Code Assistant (which is part of the Layout Painter) to generate adapter code interactively. For Natural pages the adapter code is generated completely from the page properties and the property and event bindings that you specified previously. An adapter is generated automatically when you save the layout for the first time. It is updated each time you save the layout.

Generating the Adapter

When you save the layout, a Natural adapter is generated according to the following rules:

Location	<p>The adapter is generated into the subdirectory <i>nat</i> of your project directory.</p> <p>The name of the project directory corresponds to the project name. The location of the directory depends on the application server. See Creating an Application Designer Project.</p>
Name	<p>The name of the adapter is determined by the properties you have set. See Specifying Properties for the Natural Page.</p>
Property identifiers	<p>For each control property that has been bound to an identifier (as described in Binding Properties and Methods) a parameter in the parameter data area of the adapter is generated. The identifier is therefore validated against the Natural naming conventions for user-defined variables and translated to upper-case. If an identifier does not comply to these rules, a warning is generated into the protocol and as a comment into the adapter code. Additionally, the name must comply to the naming conventions for XML entities. This means especially that the name must start with a character.</p> <p>To achieve uniqueness within 32 characters, the last four characters are (if necessary) replaced by an underscore, followed by a three-digit number.</p>
Event identifiers	<p>For each event that can be raised by a control on the page, an event handler skeleton is generated as a comment into the adapter.</p>

Caution: Some controls raise events whose names are dynamically constructed at runtime. For these events, no handler skeleton can be generated. The control reference contains information about these additional events.

The event identifiers are not validated.

Data Type Mapping

Several Application Designer controls have properties for which a data type can be specified. An example is the FIELD control. It has a `valueprop` property which can be restricted to a certain data type. The data type is used at runtime to validate user input. At generation time (that is, when a Natural adapter is generated for the page), the data type determines the Natural data format of the corresponding adapter parameter.

The following table lists the data types used in Application Designer and the corresponding Natural data formats.

Application Designer	Natural
color	A or U (depending on the NATPAGE property <code>natsinglebyte</code>). The string must contain an RGB value, for instance "#FF0000" for the color red.
date	D (YYYYMMDD)
float	F4
int	I4
long	P19
time	T (HHIISS)
timestamp	T (YYYYMMDDHHIISST)
N <i>n.n</i>	N <i>n.n</i>
P <i>n.n</i>	P <i>n.n</i>
string (default)	A or U dynamic (depending on the NATPAGE property <code>natsinglebyte</code>).
string <i>n</i>	A <i>n</i> or U <i>n</i> (depending on the NATPAGE property <code>natsinglebyte</code>).
xs:double	F8
xs:byte	I1
xs:short	I2

15

Developing the Application Code

■ Importing the Adapter	80
■ Creating the Main Program	84
■ Structure of the Main Program	86
■ Handling Page Events	86
■ Built-in Events and User-defined Events	87
■ Sending Events to the User Interface	87
■ Using Pop-Up Windows	88
■ Using Natural Maps	90
■ Navigating between Pages and Maps	90
■ Using Pages and Maps Alternatively	91
■ Starting a Natural Application from the Logon Page	92
■ Starting a Natural Application with a URL	92

Natural for Ajax Tools, which is an optional plug-in for Natural Studio, allows you to use some of the Natural for Ajax functionality which is described in this chapter directly from within Natural Studio. For further information, see *Natural for Ajax Tools* in the *Natural Studio Extensions* documentation which is provided for Natural for Windows.

Importing the Adapter

After having generated the adapter, the next step is making it available to your Natural development project.

As described previously, the adapter code is generated into a directory in your application server environment. The way you access the adapter depends on the Natural development tool you use.

The following topics are covered below:

- [Importing the Adapter Using Natural Studio](#)
- [Importing the Adapter Using Natural for Eclipse](#)

Importing the Adapter Using Natural Studio

It is assumed that your development library is located on a Natural development server and that you have mapped this development server in Natural Studio.

► To import the adapter from a remote environment

- Use drag-and-drop.

Or:

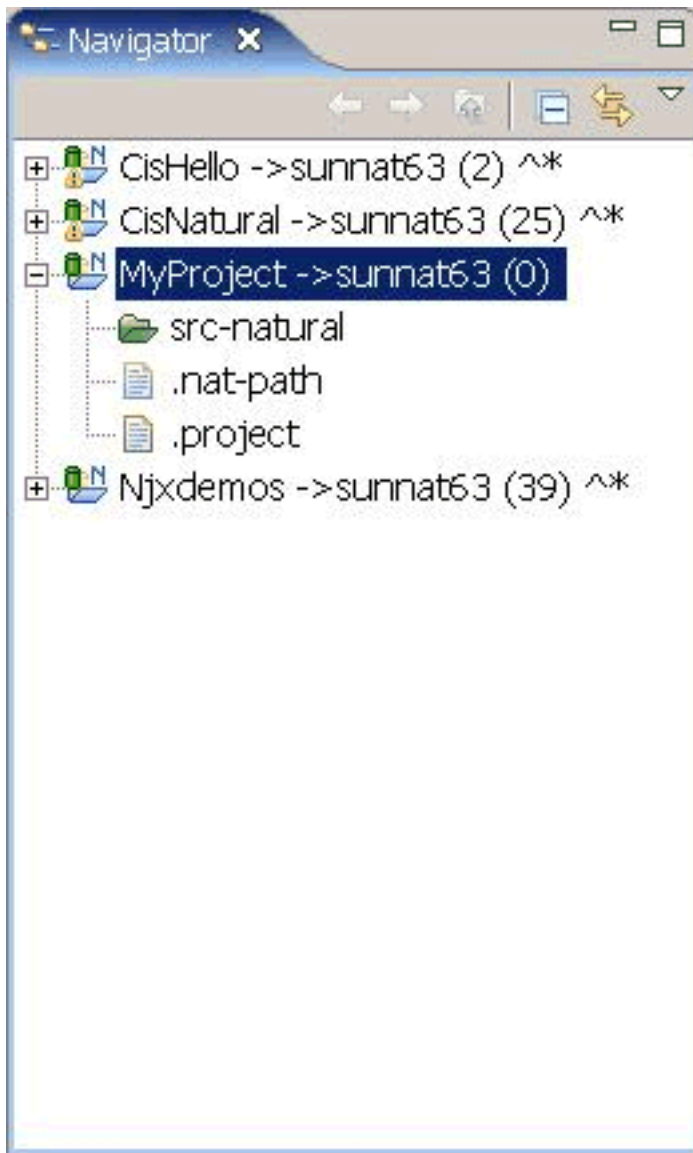
Remote UNIX environment only: Use the import function of `SYSMAIN`.

Importing the Adapter Using Natural for Eclipse

It is assumed that you have

- installed Natural for Eclipse,
- installed Application Designer's Eclipse plug-in,
- created a Natural project in Eclipse,
- established a target for the Natural project (a Natural development server).

The Navigator view will then look similar to the following:

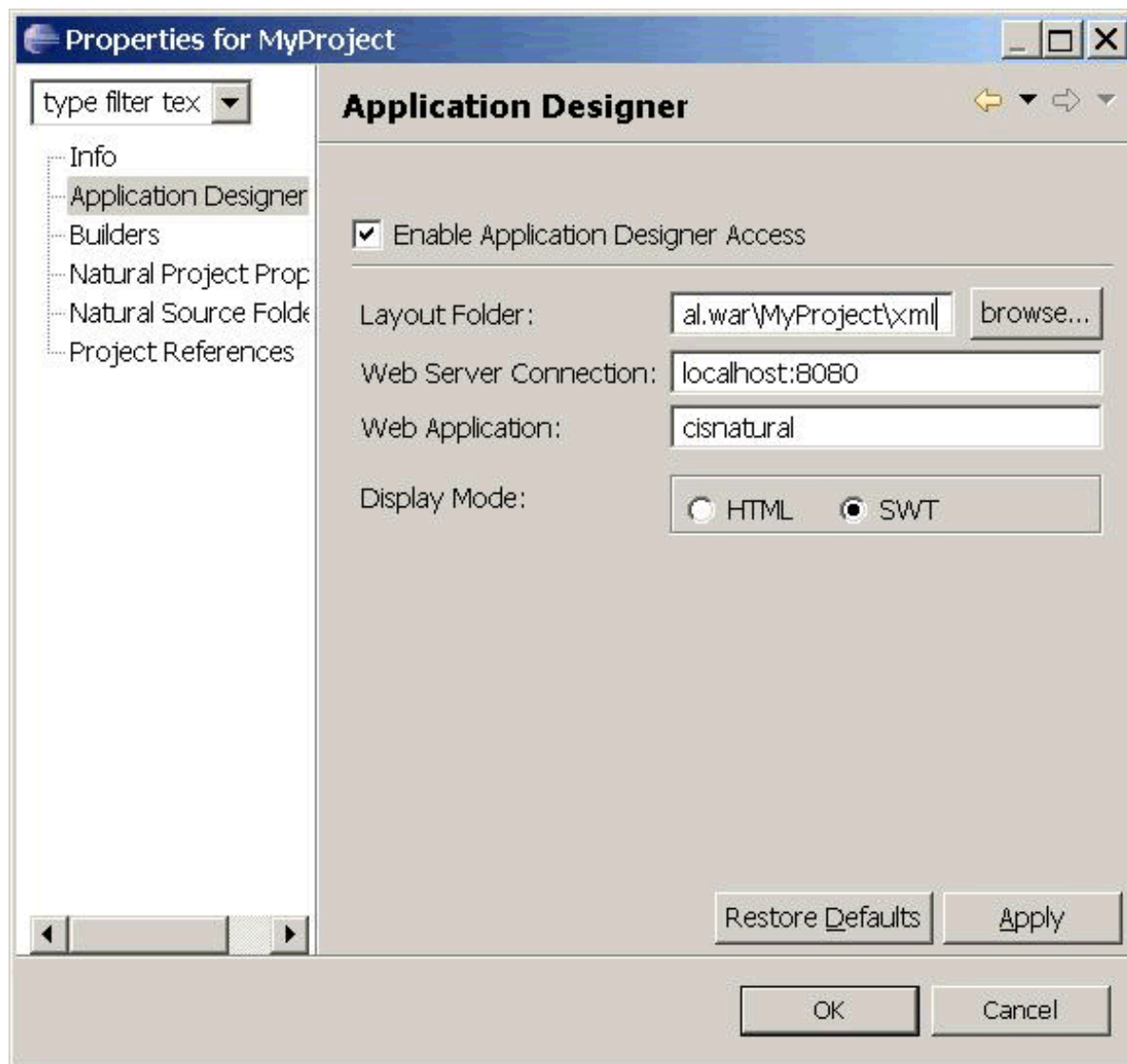


► To import the adapter from a remote environment

- 1 Proceed as described below to create the **Page Layouts** folder in your Natural project. This is the folder where you edit your page layouts with Application Designer.
 1. Invoke the **Properties** dialog for your Natural project.

2. Set the Application Designer properties as follows:

Option	Description
Layout Folder	Specify the application server directory in which the page layouts of your project are stored.
Web Server Connection	Specify host name and port number of your application server.
Web Application	Specify "cisnatural".



2. Proceed as described below to create an additional folder in your Natural project. This is the folder in which the generated adapters are located.

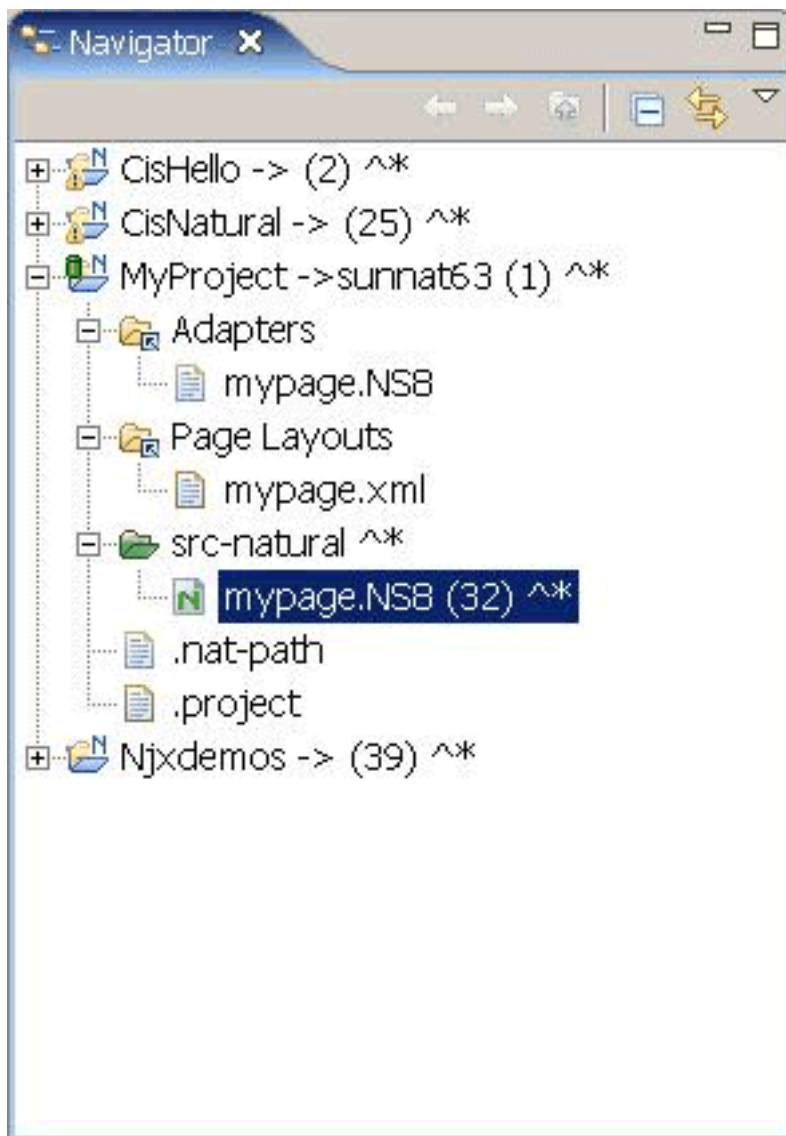
1. Select your Natural project, invoke the context menu and choose **New > Natural Folder**.

2. Expand the resulting dialog by choosing the **Advanced** button.
3. Specify a folder name of your choice (for example, "Adapters").
4. Enable the **Link to folder in the file system** check box and specify the application server directory in which the generated adapters of your project are stored.

Now you have access to your page layouts and adapters in your Natural project.

- 3 Copy or move the generated adapter from the new folder you have just created into your Natural source folder.

The Navigator view should now look similar to the following (with the new folders for the page layouts and adapters, and with your adapter in the Natural source folder).



- 4 Catalog or stow the adapter in the Natural source folder. To do so, you have to upload and compile the adapter with Natural for Eclipse.

Creating the Main Program

After you have imported the adapter, you create a program that calls the adapter to display the page and handles the events that the user raises on the page. This program can be a Natural program, subprogram, subroutine or function. We use a Natural program as example.

The adapter already contains the data structure that is required to fill the page. It contains also a skeleton with the necessary event handlers. You can therefore create a program with event handlers from an adapter in a few steps.

Open or list the adapter in the development tool of your choice (Natural Studio or Natural for Eclipse).

```
* PAGE1: PROTOTYPE      --- CREATED BY Application Designer ---
* PROCESS PAGE USING 'XXXXXXX' WITH
* FIELD1 FIELD2
DEFINE DATA PARAMETER
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE U'/MyProject/mypage' WITH
PARAMETERS
  NAME U'field1'
  VALUE FIELD1
  NAME U'field2'
  VALUE FIELD2
END-PARAMETERS
*
* TODO: Copy to your calling program and implement.
/*/*( DEFINE EVENT HANDLER
* DECIDE ON FIRST *PAGE-EVENT
*   VALUE U'nat:page.end'
*   /* Page closed.
*   IGNORE
*   VALUE U'onExit'
*   /* TODO: Implement event code.
*   PROCESS PAGE UPDATE FULL
*   NONE VALUE
*   /* Unhandled events.
*   PROCESS PAGE UPDATE
* END-DECIDE
/*/*) END-HANDLER
```

```
*
END
```

Create a new program, copy the adapter source into the program and then proceed as follows:

- Remove the comment lines in the header.
- Change `DEFINE DATA PARAMETER` into `DEFINE DATA LOCAL`.
- Replace the `PROCESS PAGE` statement with a `PROCESS PAGE USING operand4` statement, where *operand4* stands for the name of your adapter.
- Remove the comment lines that surround the `DECIDE` block.
- Uncomment the `DECIDE` block.

Your program should now look as follows:

```
DEFINE DATA LOCAL
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'MYPAGE'
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
  VALUE U'onExit'
  /* TODO: Implement event code.
  PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*
END
```

Stow the program with a name of your choice. The resulting program can be executed in a browser where it displays the page. However, it does not yet do anything useful, because it handles the incoming events only in a default way and contains no real application logic.

Structure of the Main Program

The main program that displays the page and handles its events has the following general structure:

- A `PROCESS PAGE USING` statement with the page adapter. The `PROCESS PAGE` statement displays the page in the user's web browser and fills it with data. Then, it waits for the user to modify the data and to raise an event.
- A `DECIDE` block with a `VALUE` clause for each event that shall be explicitly handled.
- A default event handler for all events that shall not be explicitly handled.

Each event handler does the following:

- It processes the data that has been returned from the page in the user's web browser.
- It performs a `PROCESS PAGE UPDATE FULL` statement to re-execute the previous `PROCESS PAGE USING` statement with the modified data and to wait for the next event.

The default event handler does not modify the data. It does the following:

- It performs a `PROCESS PAGE UPDATE` statement to re-execute the previous `PROCESS PAGE USING` statement and to wait for the next event.

Handling Page Events

When the `PROCESS PAGE` statement receives an event, the data structure that was passed to the adapter is filled with the modified data from the page and the system variable `*PAGE-EVENT` is filled with the name of the event. Now, the corresponding `VALUE` clause in the `DECIDE` statement is met and the code in the clause is executed.

The application handles the event by processing and modifying the data and resending it to the page with a `PROCESS PAGE UPDATE FULL` statement. Alternatively, it uses the `PROCESS PAGE UPDATE` statement without the `FULL` clause in order to resend the original (not modified) data.

Built-in Events and User-defined Events

There are built-in events and user-defined events.

Built-in Events

The following built-in events can be received from the page:

nat:page.end

This event is raised when the user closes the page with the Close button in the upper right corner of the page, opens another page or closes the web browser.

nat:page.default

This event is sent if the Natural for Ajax client needs to synchronize the data displayed on the page with the data held in the application. It is usually handled in the default event handler and just responded with a `PROCESS PAGE UPDATE`.

Other built-in events can be sent by specific controls. These events are described in the control reference.

User-defined Events

User-defined events are those events that the user has assigned to controls while designing the page layout with the Layout Painter. The names of these events are freely chosen by the user. The meaning of the events is described in the control reference.

Sending Events to the User Interface

The `PROCESS PAGE UPDATE` statement can be accompanied by a `SEND EVENT` clause. With the `SEND EVENT` clause, the application can trigger certain events on the page when resending the modified data.

The following events can be sent to the page:

nat:page.message

This event is sent to display a text in the status bar of the page. It has the following parameters:

Name	Format	Value
type	A or U	Sets the icon in the status bar ("S"=success icon, "W"=warning icon, "E"=error icon).
short	A or U	Short text.
long	A or U	Long text.

nat:page.valueList

This event is sent to pass values to a FIELD control with value help on request (see also the description of the [FIELD](#) control in the control reference). It has the following parameters:

Name	Format	Value
id	A or U	A list of unique text identifiers displayed in the FIELD control with value help. The list must be separated by semicolon characters.
text	A or U	A list of texts displayed in the FIELD control with value help. The list must be separated by semicolon characters.

nat:page.xmlDataMode

This event is sent to switch several properties of controls on the page in one call to a predefined state. The state must be defined in an XML file that is expected at a specific place. See the information on XML property binding in the Application Designer documentation for further information.

Name	Format	Value
data	A or U	Name of the property file to be used.

Using Pop-Up Windows

A rich GUI page can be displayed as a modal pop-up in a separate browser window. A modal pop-up window can open another modal pop-up window, thus building a window hierarchy. If a `PROCESS PAGE` statement and its corresponding event handlers are enclosed within a `PROCESS PAGE MODAL` block, the corresponding page is opened as a modal pop-up window.

The application can check the current modal pop-up window level with the system variable `*PAGE-LEVEL`. `*PAGE-LEVEL = 0` indicates that the application code is currently dealing with the main browser window. `*PAGE-LEVEL > 0` indicates that the application code is dealing with a modal pop-up window and indicates the number of currently stacked pop-up windows.

In order to modularize the application code, it makes sense to place the code for the handling of a modal pop-up window and the enclosing `PROCESS PAGE MODAL` block in a separate Natural module, for instance, a subprogram. Then the pop-up window can be opened with a `CALLNAT` statement and can thus be reused in several places in the application.

Example program `MYPAGE-P`:

```

DEFINE DATA LOCAL
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
PROCESS PAGE USING 'MYPAGE-A'
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
  VALUE U'onPopup'
  /* Open a pop-up window with the same fields.
  CALLNAT 'MYPOP-N' FIELD1 FIELD2
  PROCESS PAGE UPDATE FULL
  NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*
END

```

Example subprogram `MYPOP-N`:

```

DEFINE DATA PARAMETER
1 FIELD1 (U) DYNAMIC
1 FIELD2 (U) DYNAMIC
END-DEFINE
*
/* The following page will be opened as pop-up.
PROCESS PAGE MODAL
*
PROCESS PAGE USING 'MYPOP-A'
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
  NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*

```

```
END-PROCESS
*
END
```

Using Natural Maps

Rich internet applications written with Natural for Ajax need not only consist of rich GUI pages, but may also use classical maps. This is especially useful when an application that was originally written with maps shall only be partly changed to provide a rich GUI. In this case the application can run under Natural for Ajax from the very beginning and can then be “GUIified” step by step.

Navigating between Pages and Maps

Due to the similar structure of programs that use maps and programs that use adapters, it is easy for an application to leave a page and open a map, and vice versa. For each rich GUI page, you write a program that displays the page and handles its events. For each map, you write a program that displays the map and handles its events. In an event handler of the page, you call the program that handles the map. In an “event handler” of the map, you call the program that handles the page.

Example for program MYPAGE-P:

```
DEFINE DATA LOCAL
1 FIELD1 (U20)
1 FIELD2 (U20)
END-DEFINE
*
PROCESS PAGE USING 'MYPAGE'
*
DECIDE ON FIRST *PAGE-EVENT
  VALUE U'nat:page.end'
  /* Page closed.
  IGNORE
  VALUE U'onDisplayMap'
  /* Display a Map.
  FETCH 'MYMAP-P'
  NONE VALUE
  /* Unhandled events.
  PROCESS PAGE UPDATE
END-DECIDE
*
END
```


Example for program MYMAP-P:

```

DEFINE DATA LOCAL
1 FIELD1 (U20)
1 FIELD2 (U20)
END-DEFINE
*
SET KEY ALL
INPUT USING MAP 'MYMAP'
*
DECIDE ON FIRST *PF-KEY
  VALUE 'PF1'
  /* Display a rich GUI page.
  FETCH 'MYPAGE-P'
  NONE VALUE
  REINPUT WITH TEXT
  'Press PF1 to display rich GUI page.'
END-DECIDE
*
END

```

Using Pages and Maps Alternatively

An application can also decide at runtime whether to use maps or rich GUI pages, depending on the capabilities of the user interface. The system variable `*BROWSER-IO` lets the application decide if it is running in a web browser at all. If this is the case, the system variable tells whether the application has been started under Natural for Ajax and may thus use both maps and pages, or whether it has been started under the Natural Web I/O Interface and may thus use only maps.

Example:

```

DEFINE DATA LOCAL
1 FIELD1 (U20)
1 FIELD2 (U20)
END-DEFINE
*
IF *BROWSER-IO = 'RICHGUI'
  /* If we are running under Natural for Ajax,
  /* we display a rich GUI page.
  PROCESS PAGE USING 'MYPAGE'
  DECIDE ON FIRST *PAGE-EVENT
    VALUE U'nat:page.end'
    /* Page closed.
    IGNORE
  NONE VALUE
    /* Unhandled events.
    PROCESS PAGE UPDATE

```

```
END-DECIDE
ELSE
  /* Otherwise we display a map.
  SET KEY ALL
  INPUT USING MAP 'MYMAP'
  DECIDE ON FIRST *PF-KEY
    VALUE 'PF1'
      /* Map closed.
      IGNORE
    NONE VALUE
      REINPUT WITH TEXT
      'Press PF1 to terminate.'
END-DECIDE
END-IF
*
END
```

Starting a Natural Application from the Logon Page

In order to start a Natural application from the logon page, you proceed as described in *Configuring the Client* which is part of the *Natural Web I/O Interface* documentation.

Starting a Natural Application with a URL

See *Starting a Natural Application with a URL* and *Wrapping a Natural for Ajax Application as a Servlet* in the section *Configuring the Client* which is part of the *Natural Web I/O Interface* documentation.

16

Deploying the Application

■ Components of a Natural for Ajax Application	94
■ Unloading Natural Modules	94
■ Unloading the User Interface Components	94
■ Installing the Natural Modules	95
■ Installing the User Interface Components	95

Components of a Natural for Ajax Application

A Natural for Ajax application consists of two parts that are usually installed on two different machines.

On one hand, there are Natural modules (adapters, programs, subprograms and other Natural objects) that are installed on a Natural server. On the other hand, there are page layouts of rich GUI pages and related files that are installed in a Natural for Ajax environment on an application server.

Unloading Natural Modules

The Natural modules that belong to your application are contained in one or several Natural libraries in your Natural development environment. Unload them into a file, using the Object Handler.

Unloading the User Interface Components

The user interface components of your application are contained in one or several Application Designer projects in your Natural for Ajax development environment on your development application server.

All files in your Application Designer project are stored in one directory on the application server on which Natural for Ajax is installed. The name of the directory corresponds to the project name you have chosen. The location of the directory depends on the application server:

- **JBoss Application Server**

<installdir>/server/default/deploy/njx<nnn>.ear/cisnatural.war

- **Sun Java System Application Server**

<installdir>/domains/domain1/applications/j2ee-apps/njx<nnn>.ear/cisnatural_war

where *<installdir>* is the directory in which your application server is installed and *<nnn>* is the current Natural for Ajax version.

The project directory contains a number of subdirectories, only some of which need to be deployed to the production environment. *<projectdir>* in the table below stands for the name of your project directory. Pack the following files and subdirectories into an archive, using an archiving tool like WinZip or tar.

File	Description
<code><projectdir>/*.html</code>	Generated HTML pages.
<code><projectdir>/xml/*.*</code>	Page layouts.
<code><projectdir>/wsdl/*.*</code>	Page data schemas.
<code><projectdir>/accesspath/*.*</code>	Page data access definitions.
<code><projectdir>/multilanguage/*.*</code>	Language-dependent strings.
<code><projectdir>/help/*.*</code>	Language-dependent help texts.

Installing the Natural Modules

In order to install the Natural modules in the production environment, load them with the Object Handler.

Installing the User Interface Components

In order to install the user interface components, unpack the previously created archive into a corresponding project directory in your Natural for Ajax production environment on your production application server.

17

Natural Parameters and System Variables

The following Natural parameters and system variables are evaluated in Natural for Ajax applications and sent to Application Designer:

- DC

The character assigned to the DC parameter is used in the representation of decimal fields in Application Designer.

- DTFORM

This parameter is used for all date fields in Application Designer pages. In your application, the date is shown according to the setting of the DTFORM parameter.

- *CURS-FIELD

Identify the operand that represents the value of the control that has the input focus. When the Natural system function POS is applied to a Natural operand that represents the value of a control, it yields the identifier of that operand.

- *LANGUAGE

Change the language while an application is running. See also [Multi Language Management](#).

See also [Support for Special Features](#).

18

Usage of Edit Masks

■ General Information	100
■ Data Types with Edit Masks	100
■ Characters Used in Edit Masks	100
■ Specifying Edit Masks in Layouts	101
■ Static versus Dynamic Validation	103

General Information

Natural for Ajax supports a subset of the Natural edit mask concept, in order to support output formatting and input validation of numeric fields.

If an edit mask is specified for a numeric field, the field content is rendered according to the edit mask. Input to this field is validated against a regular expression that corresponds to this edit mask. The regular expression is automatically generated from the edit mask, so that by defining the edit mask, the user controls at design time the formatting of the field as well as the validation.

During the output formatting and input validation for numeric fields with edit masks, the Natural parameters `DC`, `THSEPCH` and `EMFM` are supported.

Data Types with Edit Masks

In all controls that support the property `datatype`, an edit mask can be specified for the following data types:

- `N n.n`
- `P n.n`
- `int`
- `float`
- `xs:short`
- `xs:decimal`

Characters Used in Edit Masks

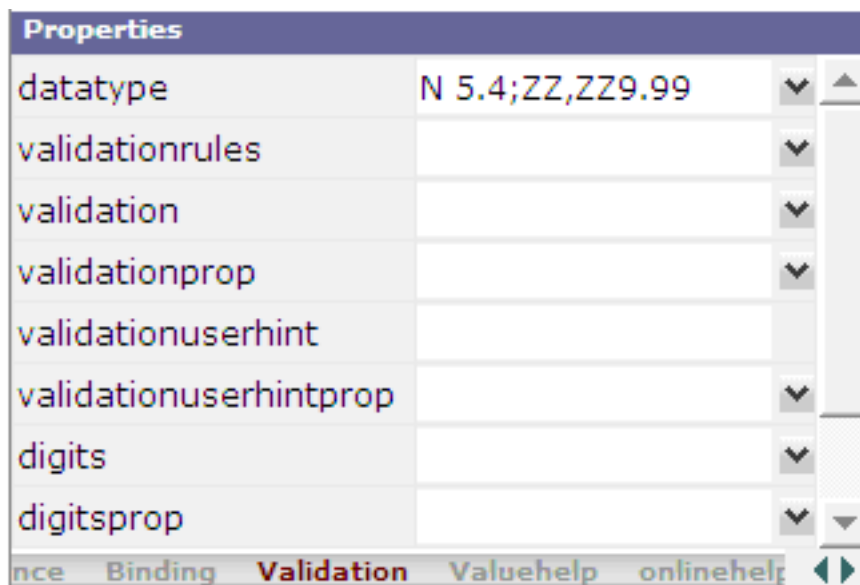
The following characters can be used in edit masks:

Character	Function
9	Decimal digit or digit.
Z	Zero-suppressed digit.
+	Preceding/following sign.
-	Preceding/following sign if the value is negative.

Character	Function
,	Thousands separator character. Note: The actual character used as the thousands separator can be either a comma (,) or a period (.). This depends on the setting of the parameter THSEPC in Natural.
.	Decimal separator character. Note: The actual character used as the decimal separator can be either a period (.) or a comma (,). This depends on the setting of the parameter DC in Natural.

Specifying Edit Masks in Layouts

An edit mask is added to a numeric data type in the following way (the edit mask is separated from the data type by a semicolon):



Application Designer generates a regular expression from the edit mask and adds it as a value for the `validation` property. This regular expression is used to validate the value on the client side when the user changes the field. This is done in the same way as if the user had specified the regular expression directly in the `validation` property.

The generated regular expression can be seen in the generation protocol. For the above example, it is shown as follows:

Id ▢	Tag ▢	Type ▢	Message
20	itr		
21	label		name/N 5.4;ZZ,ZZ9.99 width/150
22	field		valueprop/thdecsepfield width/200 flush/server datatype/N 5.4;ZZ,ZZ9.99
		Info	Validation added [0-9,.]{1,6}[.]{1}+[0-9]{1,2}

The validation takes place *after* the modification of a field value, that is: when the focus has left the field. This means that the validation does not take place with every key pressed. The user can enter invalid values as long as the input focus is in the field.

Input Rendering

For ZZ,ZZ9.99 of the above example, the user can enter a valid value even without entering the thousands separator character:

Edit Masks With Decimal Separator ▾
N 5.4;ZZ,ZZ9.99

Since the value itself is correct, it is accepted. Ideally, the value should be rendered according to the precise edit mask when the focus of the field is left. This can be easily achieved by setting the `flush` property to "server". This will trigger a server roundtrip and since output rendering takes place with each server roundtrip, the rendering is done automatically.

Edit Masks With Decimal Separator ▾
N 5.4;ZZ,ZZ9.99

Input Completion

Application Designer does some input completion, for example, when working with `float` data types. An example for this is adding "0" decimal digits.

For the above input, Application Designer will do the following input completion:

Static versus Dynamic Validation

At design time, the characters that will actually be used for `DC` and `THSEPCH` are not yet known. Therefore, when the regular expression for validation is already generated at design time, both characters must be considered as possible `DC` or `THSEPCH` characters. In consequence, the validation is more tolerant than it should be.

To achieve a more stringent validation, the regular expression can also be dynamically generated at runtime. This is achieved by appending the literal "dynamic" to the edit mask, separated by a semicolon. In this case, the runtime settings of `DC` and `THSEPCH` are evaluated during the validation. Example:

```
datatype="N 5.4;+ZZ,999.99;dynamic"
```

If the parameter `EMFM` is to be evaluated, the validation expressions must be generated dynamically, because at design time, it is not yet known whether the application will run in free mode or not. Statically generated validations always behave as in free mode. That is: they do not force the user to enter plus or minus signs or the thousands separator character.

19

Multi Language Management

The multi language management is responsible for changing the text IDs into strings that are presented to the user.

There are two translation aspects:

- All literals in the GUI definitions of a layout are replaced by strings which are language-specific. This is based on the multi language management of Application Designer.



Note: Detailed information on the multi language management is provided in the Application Designer documentation at http://documentation.softwareag.com/webmethods/cit_reroute.htm.

- Literals that are contained in your application code are handled with the language management of Natural.

In a Natural for Ajax application, both language management systems are related by common language codes. The language codes used are those that are defined for the Natural profile parameter `ULANG` and the system variable `*LANGUAGE`.

The Application Designer documentation describes how the text files containing the language-dependent texts are created and maintained (see the information on writing multi language layouts at the above URL). For a multi-lingual Natural for Ajax application, the names of the directories that contain the text files should be chosen according to the Natural language codes, for instance `/multilanguage/4` for Spanish texts.

When an application is started from the Natural logon page (see [Starting a Natural Application from the Logon Page](#)), the user can select the language to be used. Depending on the selected language, the same (Natural) language code is set up both in Application Designer and in the Natural session, so that both language management systems are then configured to use the same language.



Note: The language for a session can also be defined in the configuration file *sessions.xml*, with the element `language`. See *Managing the Configuration File for the Session* in the *Natural Web I/O Interface* documentation.

It is also possible to change the language while an application is running. This is done by setting the Natural system variable `*LANGUAGE` in the Natural program. Each time this system variable is changed, Natural for Ajax changes the language code for the web pages when the next update of the page occurs.

For compatibility with the predefined multi language directories in Application Designer, the English and German texts need not be stored in */multilanguage/1* and */multilanguage/2*, but can be contained in */multilanguage/en* and */multilanguage/de*.

20 Support of Right-to-Left Languages

Natural for Ajax supports right-to-left languages and bidirectional text without specific actions taken by the application. The browser displays and accepts bidirectional text always in the expected order.

Applications can use the same page layouts both in left-to-right and in right-to-left screen direction. To switch the screen direction, the statement `SET CONTROL` is used as follows:

Statement	Description
<code>SET CONTROL 'VON'</code>	Sets the screen direction to right-to-left.
<code>SET CONTROL 'VOFF'</code>	Sets the screen direction to left-to-right.
<code>SET CONTROL 'V'</code>	Switches from left-to-right to right-to-left screen direction and vice versa.

21

Server-Side Scrolling and Sorting

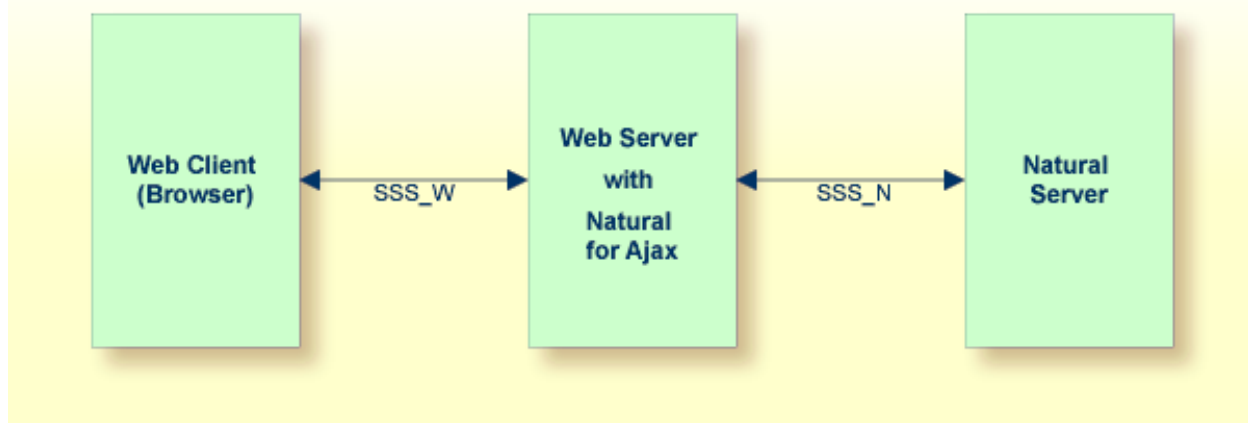
■ General Information	110
■ Variants of Server-Side Scrolling and Sorting	110
■ Controls that Support Server-Side Scrolling and Sorting	114
■ Data Structures for Server-Side Scrolling and Sorting	114
■ Server-Side Scrolling and Sorting in Trees	116
■ Events for Server-Side Scrolling and Sorting	117

General Information

It is often the case that a web application has to display an arbitrary amount of data in a grid control, for instance, the records from a database table. In these cases, it is mostly not efficient to send all data as a whole to the web client. Instead, it will be intended to display a certain amount of data to begin with and to send more data as the user scrolls through the page. To support this, the grid controls in Natural for Ajax support the concept of server-side scrolling and sorting.

Variants of Server-Side Scrolling and Sorting

The following graphic illustrates the different types of server-side scrolling and sorting that are supported by Natural for Ajax.



With respect to server-side scrolling and sorting, the following options can be used:

■ No Server-Side Scrolling and Sorting

The Natural application sends the grid data to the web server as a whole. The web server sends the grid data to the web client (browser) as a whole.

Advantage: Neither the web server nor the Natural application are involved in the process of scrolling and sorting. As long as the user only scrolls and sorts, no round trip from the web client to the web server or to the Natural server is necessary.

Disadvantage: A round trip between web server and Natural server that is triggered by other user actions transports the entire grid data.

■ Web Server-Side Scrolling and Sorting (SSS_W)

The Natural application sends the grid data to the web server as a whole. The web server sends the grid data to the web client (browser) in portions.

Advantage: The Natural application is not involved in the process of scrolling and sorting. As long as the user only scrolls and sorts, no round trip from the web server to the Natural server is necessary.

Disadvantage: A round trip between web server and Natural server that is triggered by other user actions transports the entire grid data.

■ Natural Server-Side Scrolling and Sorting (SSS_N)

The Natural application sends the grid data to the web server in portions. The web server sends the grid data to the web client (browser) in portions.

Advantage: A round trip between web server and Natural application passes only the visible data portion.

Disadvantage: The Natural application must support the process of scrolling and sorting with a specific application logic.

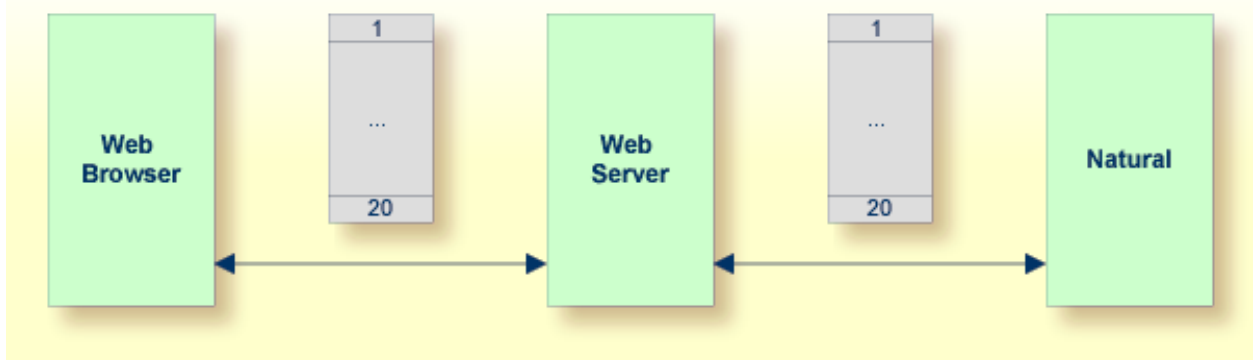
The decision between these options will often depend on the expected data volume. The application can decide dynamically at runtime which option to use.

The following topics show the difference between these three options

- [No Server-Side Scrolling and Sorting](#)
- [Web Server-Side Scrolling and Sorting](#)
- [Natural Server-Side Scrolling and Sorting](#)

No Server-Side Scrolling and Sorting

Step 1: The grid is configured at design time to a row count of twenty. The Natural application sends twenty rows and indicates that no further rows are to be expected (`SIZE=0`).

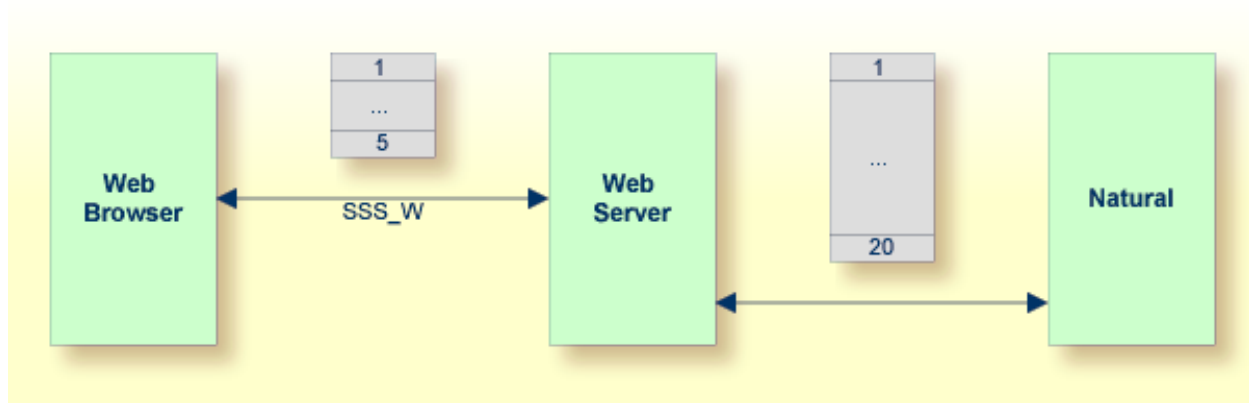


Step 2: When you scroll up and down, no server round trips to the web server or to the Natural application are performed.

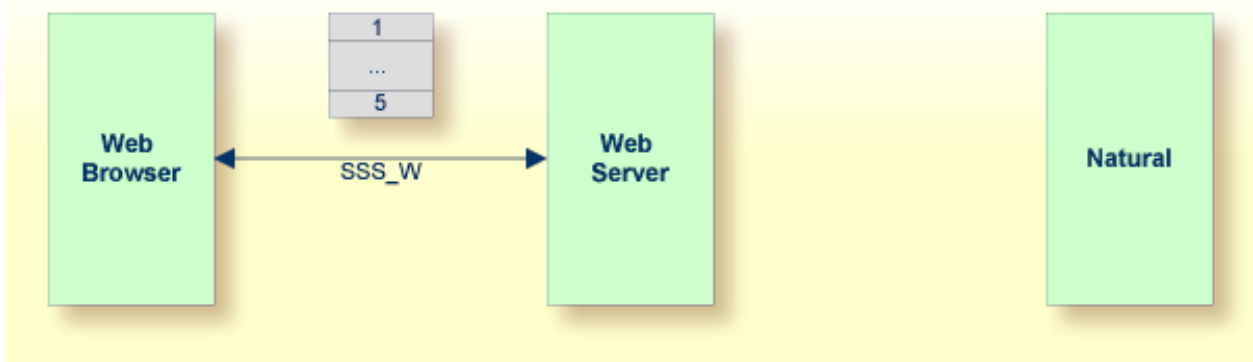


Web Server-Side Scrolling and Sorting

Step 1: The grid is configured at design time to a row count of five. The Natural application sends twenty rows and indicates that no further rows are to be expected (SIZE=0).

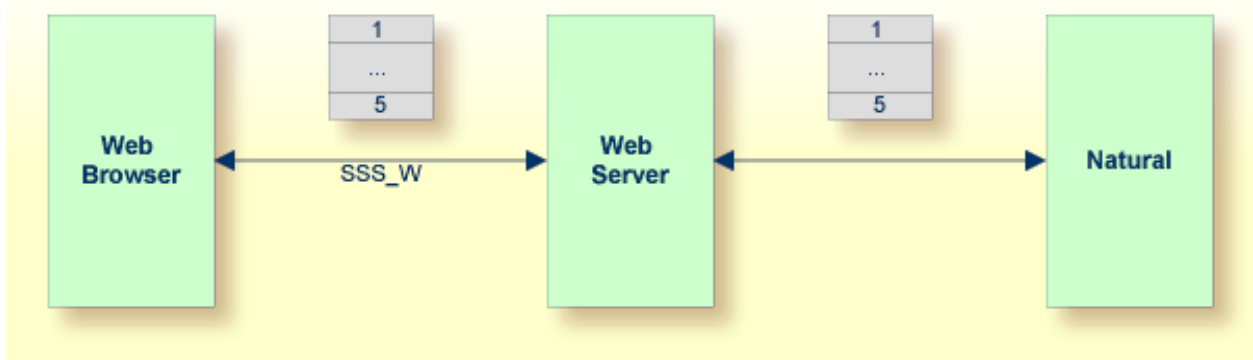


Step 2: When you scroll up and down, the web browser requests additional records from the web server. There are no server round trips to Natural.

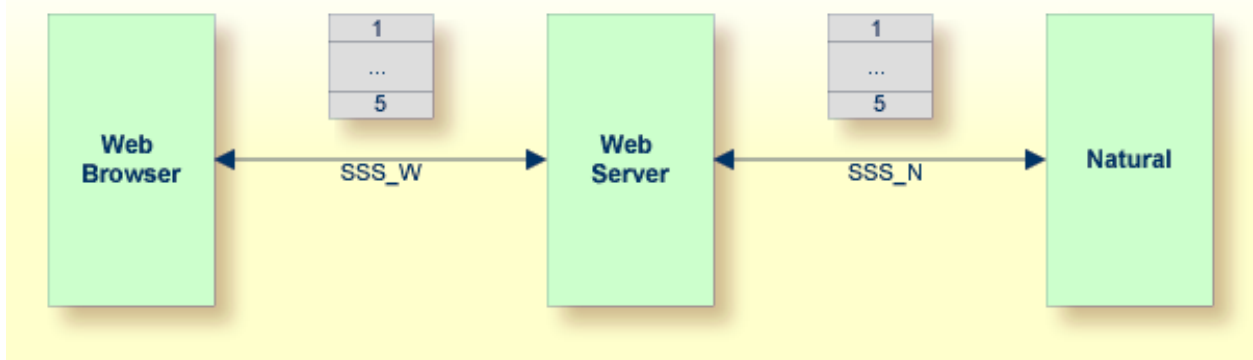


Natural Server-Side Scrolling and Sorting

Step 1: The grid is configured at design time to a row count of five. The Natural application sends five rows and indicates that further rows are to be expected (`SIZE=20`).



Step 2: When you scroll up and down, the web browser requests additional records from the web server. The web server requests additional records from the Natural application.



The Natural application can dynamically decide at runtime which option of server-side scrolling and sorting it wants to use. This can depend on the number of records contained in a search result.

- If the application does not want to use server-side scrolling and sorting at all, it sends as many rows to the web browser as the grid is configured to hold, or it sends fewer rows.
- If the application wants to use web server-side scrolling and sorting, it sends all available rows and sets the `SIZE` parameter to zero in the data structure that represents the grid in the application.
- If the application wants to use Natural server-side scrolling and sorting, it sends only part of the available rows and indicates in the `SIZE` parameter how many rows are to be expected altogether.

Controls that Support Server-Side Scrolling and Sorting

The following controls support server-side scrolling and sorting:

- **TEXTGRIDSSS2**
- **ROWTABLEAREA2**
- **MGDGRID**



Note: For compatibility reasons with earlier versions of Natural for Ajax, you have to set the `natsss` property of NATPAGE to true in order to activate server-side scrolling and sorting for the controls ROWTABLEAREA2 and MGDGRID. If this property is set to true, for all instances of these grid controls on a page, the necessary data structures are generated into the Natural adapter interface.

Data Structures for Server-Side Scrolling and Sorting

If you use the TEXTGRIDSSS2 control or if you use the ROWTABLEAREA2 or MGDGRID control and have set the property `natsss` to true for the page, the following additional data structure is generated into the adapter interface for each instance of these controls. This data structure is used to control the scroll and sort behavior at runtime.


```

1 LINESINFO
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)

```

The name of the data structure is derived from the name of the variable that is bound to the grid. In this example, the variable `LINES` had been bound to the grid. Therefore, the name `LINESINFO` was generated.

With each event that is related to scrolling and sorting, the application receives the information how many rows it should deliver at least (`ROWCOUNT`) and the index of the first record to be delivered (`TOPINDEX`).

In `SORTPROPS`, the application receives the information in which sort sequence the records should be delivered and by which columns the records should be sorted.

On the other hand, the application itself can specify a sort sequence (also using multiple sort criteria) and indicate this sort sequence by filling the structure with the desired sort criteria.

- If web server-side scrolling and sorting is used, the specified sort sequence is automatically created on the web server.
- If Natural server-side scrolling and sorting is used, the application itself must provide the records in the specified sort sequence.
- With the `TEXTGRIDSSS2` control, the first three specified sort criteria are automatically indicated in the column headers of the grid.
- With the `ROWTABLEAREA2` control, the first specified sort criterion is automatically indicated in the column headers of the grid. If more sort criteria are to be indicated, the application should provide custom grid headers.

In `SIZE`, the application can indicate whether the delivered amount of rows represents all available data (`SIZE=0`, no Natural server-side scrolling), or whether there are more rows to come (`SIZE=total-number-of-records`, Natural server-side scrolling).

When Natural server-side scrolling is used, the application will, for instance, hold the available rows (mostly the result of a database search) in an *X*-array, sort this *X*-array as requested and deliver the requested portion of rows. However, other implementations and optimizations are possible, depending on the needs and possibilities of the application.

Server-Side Scrolling and Sorting in Trees

The ROWTABLEAREA2 control can also be configured as a tree control, where each row represents a tree node. In this case, the data structure that supports server-side scrolling contains one more field, DSPINDEXFIRST.

```
1 LINESINFO
2 DSPINDEXFIRST (I4)
2 ROWCOUNT (I4)
2 SIZE (I4)
2 SORTPROPS (1:*)
3 ASCENDING (L)
3 PROPNAME (U) DYNAMIC
2 TOPINDEX (I4)
```

The need for this additional control field comes from the fact that a tree can contain hidden items.

The rows sent by the Natural application must always start with an item at level one. The additional field DSPINDEXFIRST is provided because the visible part of the tree can start at a node with a level greater than one (a subnode). In DSPINDEXFIRST, the application must indicate the index of the first visible row within the rows sent from Natural.

Example

LN	Tree Nodes	Label
1	▼ toptext_0	lineinfo_0
2	■ childtext_0.0	childlineinfo_
3	■ childtext_0.1	childlineinfo_
4	■ childtext_0.2	childlineinfo_
5	■ childtext_0.3	childlineinfo_▼

The top nodes of the tree are open and the user scrolls down as shown below:

LN	Tree Nodes	Label
4	■ childtext_0.2	childlineinfo_
5	■ childtext_0.3	childlineinfo_
6	■ childtext_0.4	childlineinfo_
7	▼ toptext_1	lineinfo_1
8	■ childtext_1.0	childlineinfo_▼

The Natural application is supposed to send data starting with a top node. In our example, this is the node named **toptext_0**. But the first visible child node would be **childtext_0.2**. This means that among the sent items, the first three items are hidden. The application sets the value for `DSPINDEXFIRST` to "3" when sending the data.

Events for Server-Side Scrolling and Sorting

In order to support server-side scrolling and sorting, an application must handle a number of related events properly. The events are described with the corresponding controls. Examples on how to handle the events are provided in the library `SYSEXNJX`.

22 Application Modernization

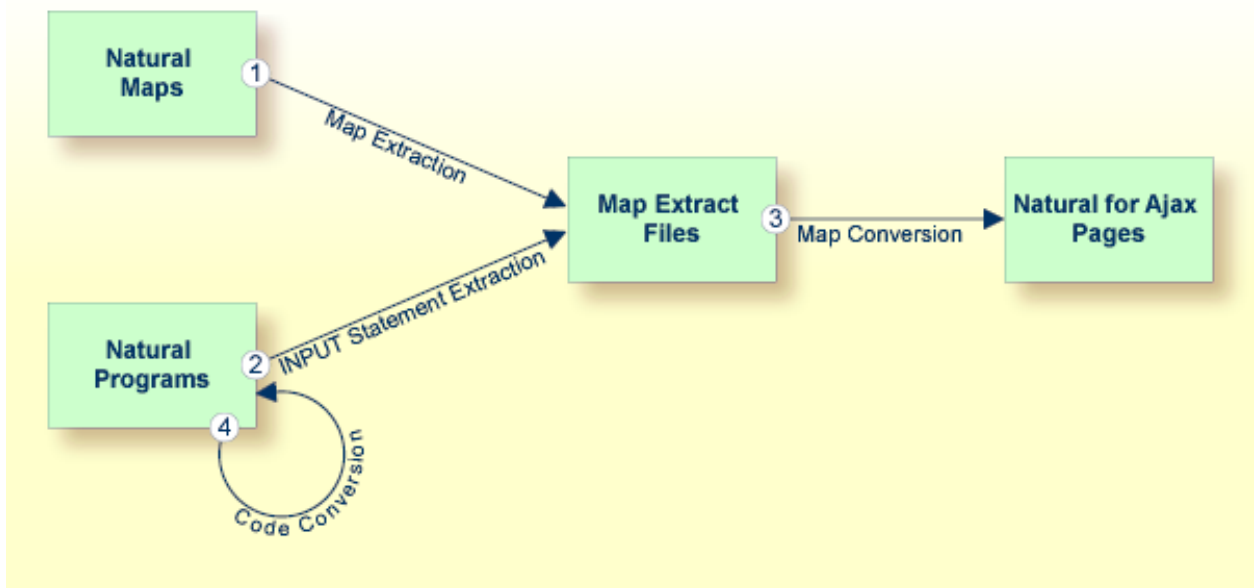
This part describes how to convert a character-based Natural application to a Natural for Ajax application.

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

- **Overview of Conversion Steps**
- **Map Extraction**
- **Map Conversion**
- **Customizing the Map Conversion Process**
- **Code Conversion**

23 Overview of Conversion Steps

The conversion of a character-based Natural application to a Natural for Ajax application consists of several steps as illustrated in the following graphic:



■ Step 1: Map Extraction

Extracts from each Natural map the information that is required to create a corresponding Natural for Ajax page. For each map, a map extract file is created. This file is intended as input for the map conversion.

Required tool: Natural Studio which is part of Natural for Windows.

See [Map Extraction](#) for further information.

■ **Step 2: INPUT Statement Extraction**

This step is required for Natural applications that do not use maps, but use `INPUT` statements for the dynamic specification of the screen layouts.

Extracts from each `INPUT` statement in the source code the information that is required to create a corresponding Natural for Ajax page. For each `INPUT` statement, a map extract file is created. This file has the same format as a map extract file created by the map extraction process, and it is also intended as input for the map conversion.

Required tool: Natural for Ajax Conversion utility which is part of Natural Engineer.

■ **Step 3: Map Conversion**

Processes the map extract files and creates the corresponding Natural for Ajax pages.

Required tool: Map Converter which is part of the Application Designer development workplace contained in Natural for Ajax.

See [Map Conversion](#) and [Customizing the Map Conversion Process](#) for further information.

■ **Step 4: Code Conversion**

This step requires that the Natural for Ajax pages have already been created.

Modifies the application code in such a way that it can use the newly created Natural for Ajax pages. The application can still run in a terminal, in the Natural Web I/O Interface client or in batch as before. But it can now also run in a Natural for Ajax session with the new Natural for Ajax pages.

Required tool: Natural for Ajax Conversion utility which is part of Natural Engineer.

Code conversion can also be performed manually. See [Code Conversion](#) for further information.

The resulting Natural for Ajax application mimics the character-based application. The user interface is not restructured in the sense that several maps are combined into a single page or that complex maps are split into several separate pages. This kind of restructuring is not part of the conversion, but of the normal development of a Natural for Ajax application.

24

Map Extraction

■ General Information	124
■ Using Natural for Ajax Tools	124
■ Using the Mass Function	124
■ Location of the Files	124

General Information

The Map Extractor is the first tool that is used in the process of converting a map-based application to a Natural for Ajax application. It analyzes the code of a Natural map and creates from each map a file that contains information about the map, the so-called “map extract file”.

The map extract files have the extension *.njx* and are not human-readable. They are intended as input for the second step of the process, the [map conversion](#).

The Map Extractor is used only to process character maps. GUI elements contained in maps are not extracted.

Using Natural for Ajax Tools

The map extract files can be created using Natural for Ajax Tools, which is an optional plug-in for Natural Studio. See *Using the Map Extractor* in the *Natural Studio Extensions* documentation which is provided for Natural for Windows.

Using the Mass Function

For mass processing of maps, the Natural program `MAP2NJX` is provided. The program is delivered in the plug-in library `SYSPLNJX`.

`MAP2NJX` is working only on the local environment. It is called in the following way:

```
MAP2NJX library-name map-name
```

In the parameter *map-name*, the asterisk (*) notation can be used.

Location of the Files

The location of the map extract files depends on the settings in the configuration file *ConfigNJX-PLG.dat* (see *Configuring the Servers* in the *Natural Studio Extensions* documentation which is provided for Natural for Windows).

If an application server and a Natural Web I/O Interface server has been specified for the active environment, *and* if a file-system path to the application server environment has been specified, *and* if an Application Designer project has been created for the current library, *and* if this Application

Designer project contains a *nat* subdirectory, then the Map Extractor writes the resulting map extract files to the *nat* subdirectory of this Application Designer project.

If the above information is not available for the active environment, the Map Extractor stores the files as follows:

- If the active environment is the local environment, the files are stored in the *res* subdirectory of the current library.
- If the active environment is a remote environment, the extraction fails.

The names of the map extract files are derived from the map names (for example, MYMAP . NSM results in MYMAP . NJX).

If the function is applied to several maps and the extraction fails for some of them, a log is displayed and is stored at the location described below:

- If the above path information is available and valid, the log file is written to the *nat* subdirectory of the corresponding Application Designer project.
- If the above path information is available, but not valid, the log file is written to the *res* subdirectory of the user's library in the local environment.
- If the above path information is not available and the active environment is a remote environment, the log file is written to the *res* subdirectory of the user's library in the local environment.
- If the above path information is not available and the active environment is the local environment, the log file is written to the *res* subdirectory of the current library.

25

Map Conversion

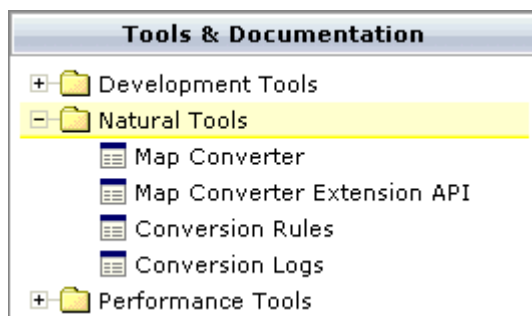
■ General Information	128
■ First Steps	129
■ Using the Map Converter	137
■ Using the Editor Extension	141
■ Using the Conversion Rules Tool	144
■ Using the Conversion Logs Tool	146

General Information

After the [Map Extractor](#) or the `INPUT` Extractor has been used to create extract files from maps, the Map Converter is the next tool used in the process of converting a map-based application into a Natural for Ajax application. The Map Converter processes the map extract files that were created by the Map Extractor or the `INPUT` Extractor. It analyzes the map extract files and creates a Natural for Ajax page layout from each map extract file. Controls on the map are converted to controls on the page. Many features of the original map are converted to features of the page.

By default, the Map Converter uses a predefined set of page templates and conversion rules that control the conversion process. The templates and the conversion rules can be modified or extended to adapt the converter to the requirements of a specific conversion project. With the advanced option to program own conversion handlers, the Map Converter provides additional flexibility and extensibility.

The Application Designer development workplace contained in Natural for Ajax provides additional Natural tools for map conversion:



The following Natural tools can be invoked from the navigation frame:

■ Map Converter

This tool is used for mass generation of layouts. For quick start with this tool, see [First Steps](#) below. For detailed information on all options of this tool, see [Using the Map Converter](#).

You can also generate a single layout while designing a page in the Layout Painter. An editor extension is available for this purpose. See [Using the Editor Extension](#) for further information.

■ Conversion Rules

You can use this tool to copy the conversion rules from other projects to the current project. See [Using the Conversion Rules Tool](#) for further information.

■ Conversion Logs

You can use this tool to view or delete the log files that have been created during the conversion. See [Using the Conversion Logs Tool](#) for further information.

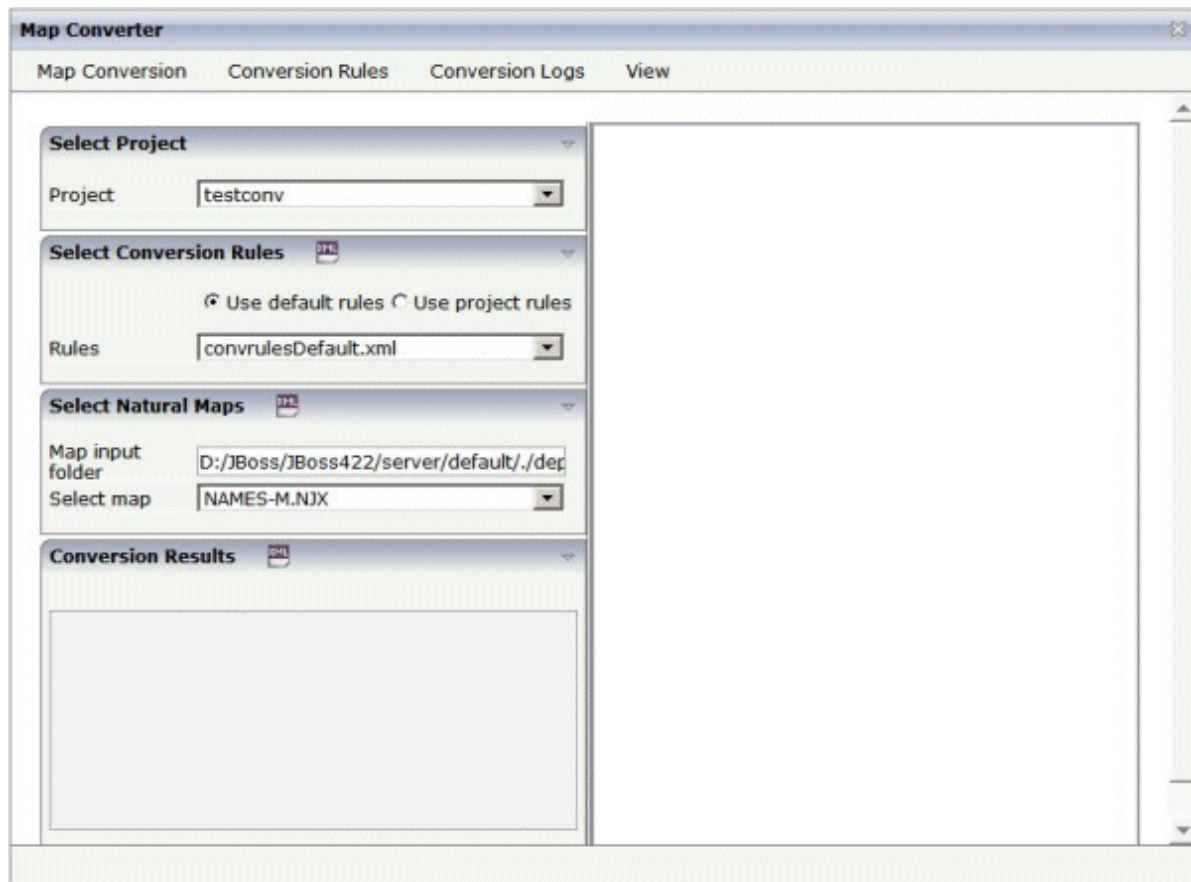
First Steps

We start with a simple map like the one below and we suppose that you have already created a map extract file with the Map Extractor. The map is contained in a Natural library named `TESTCONV`. The map extract file has been stored in the *nat* subdirectory of an Application Designer project with the corresponding name *testconv*.

► To create a Natural for Ajax page layout from an extract file

- 1 Open the Application Designer development workplace.
- 2 In the **Natural Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Map Converter**.

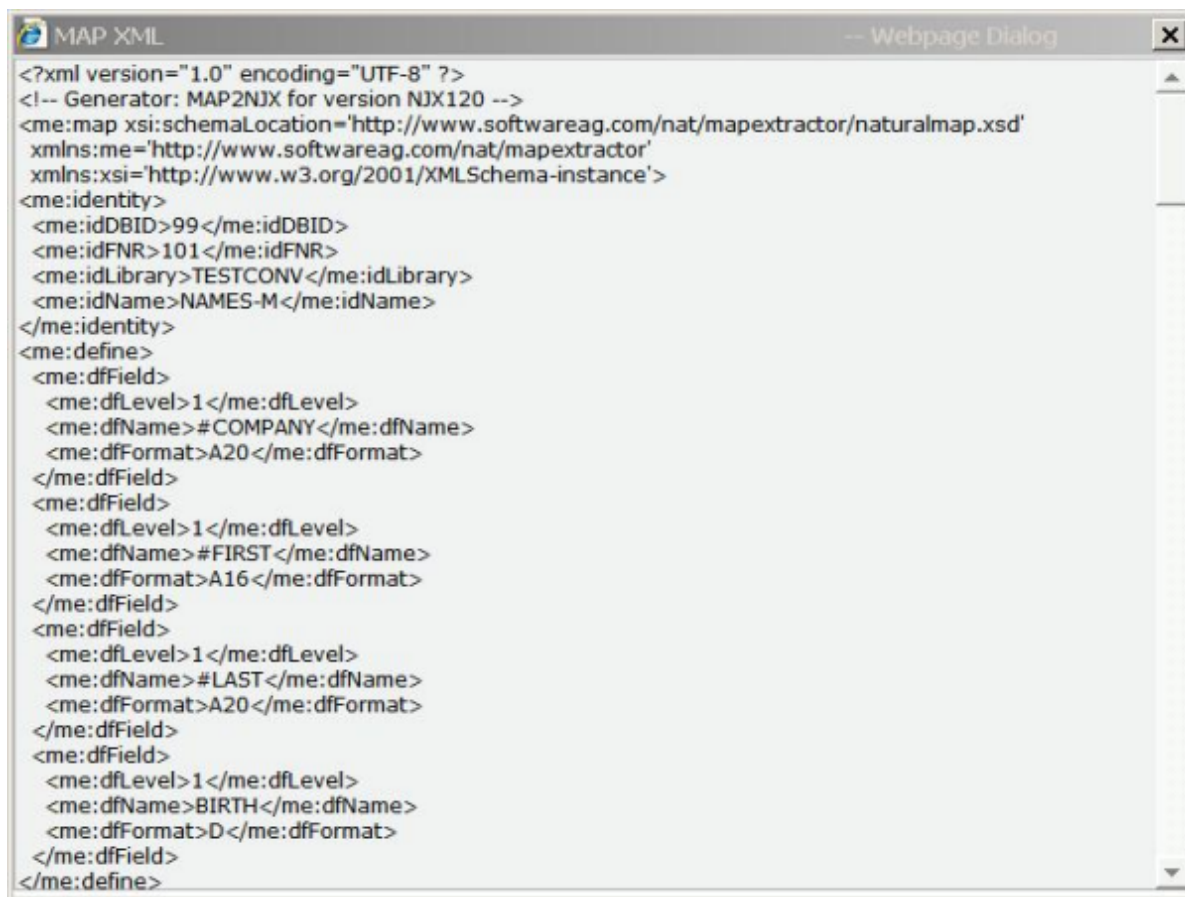
The Map Converter is opened.



- 3 Select the project in which you want to store the page layouts that are to be generated. That is, select your project *testconv*.
- 4 Select the conversion rules file to be used. That is, stick with the rules file *convrulesDefault.xml* to begin with.
- 5 Select the map input folder, that is, the folder in which your map extract files are stored.
- 6 Select a map extract file.
- 7 From the **Map Conversion** menu, choose **Show Map** to display the content of the map extract file in XML format.

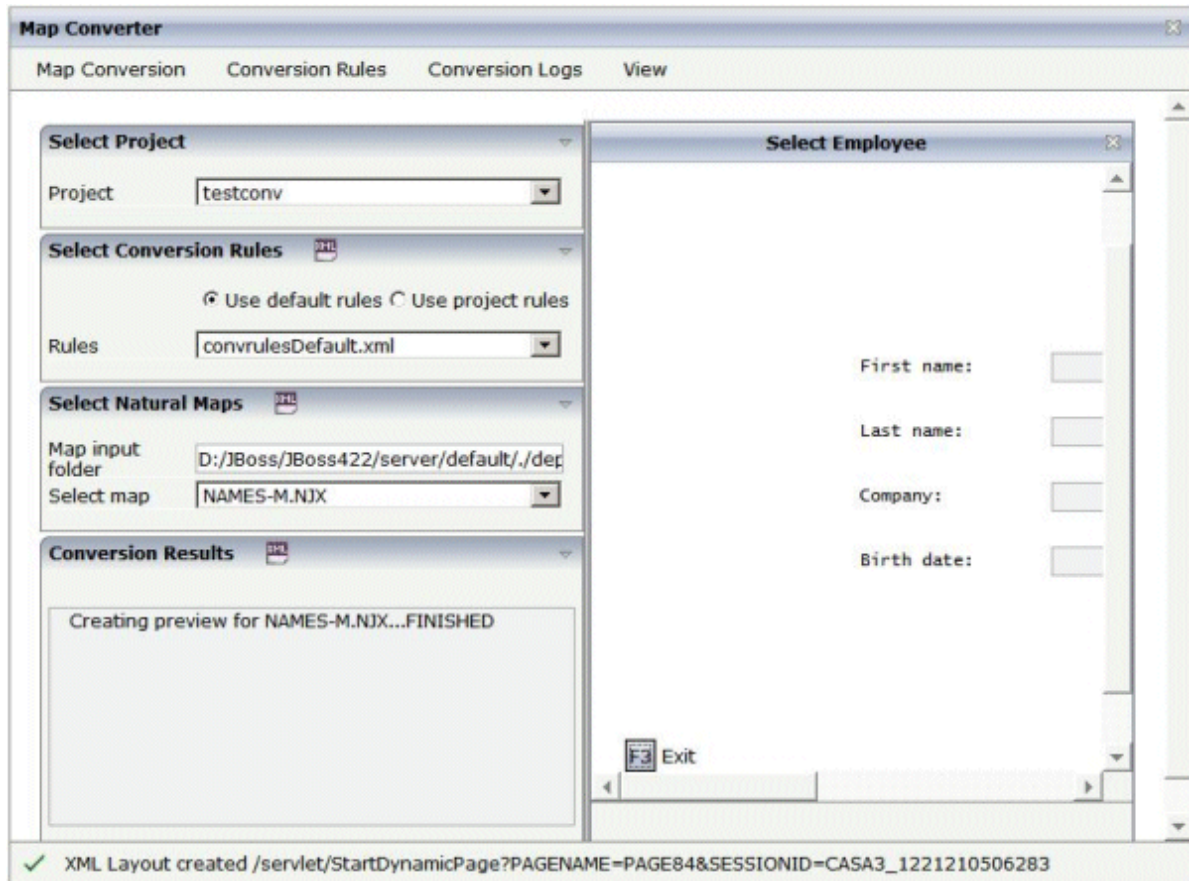
Or:

Choose the icon that is shown in the **Select Natural Maps** header.



- 8 From the **Map Conversion** menu, choose **Preview Page Layout** to display the resulting page layout as it would turn out using the selected conversion rules file.

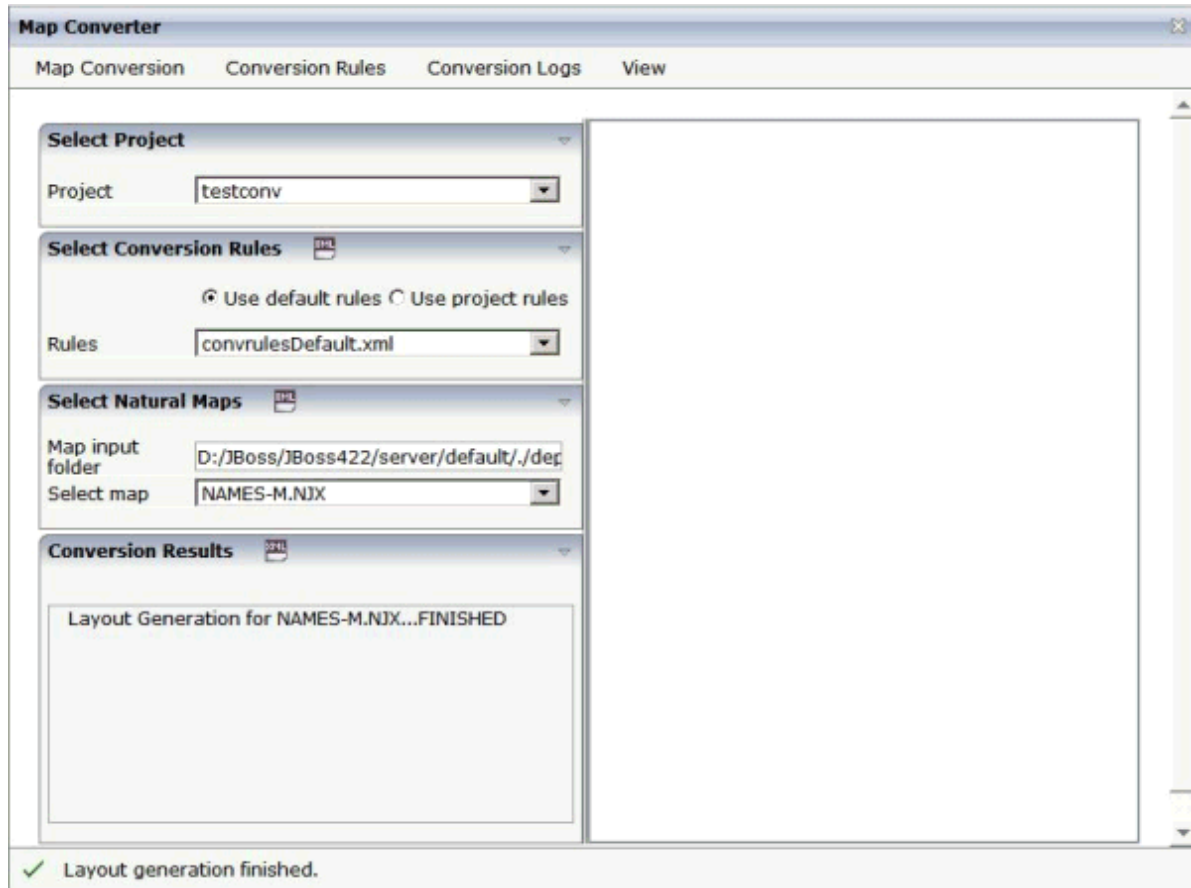
The right side shows a preview of the generated page layout. The **Conversion Results** area shows a status message which informs either about successful conversion or an error that has occurred.



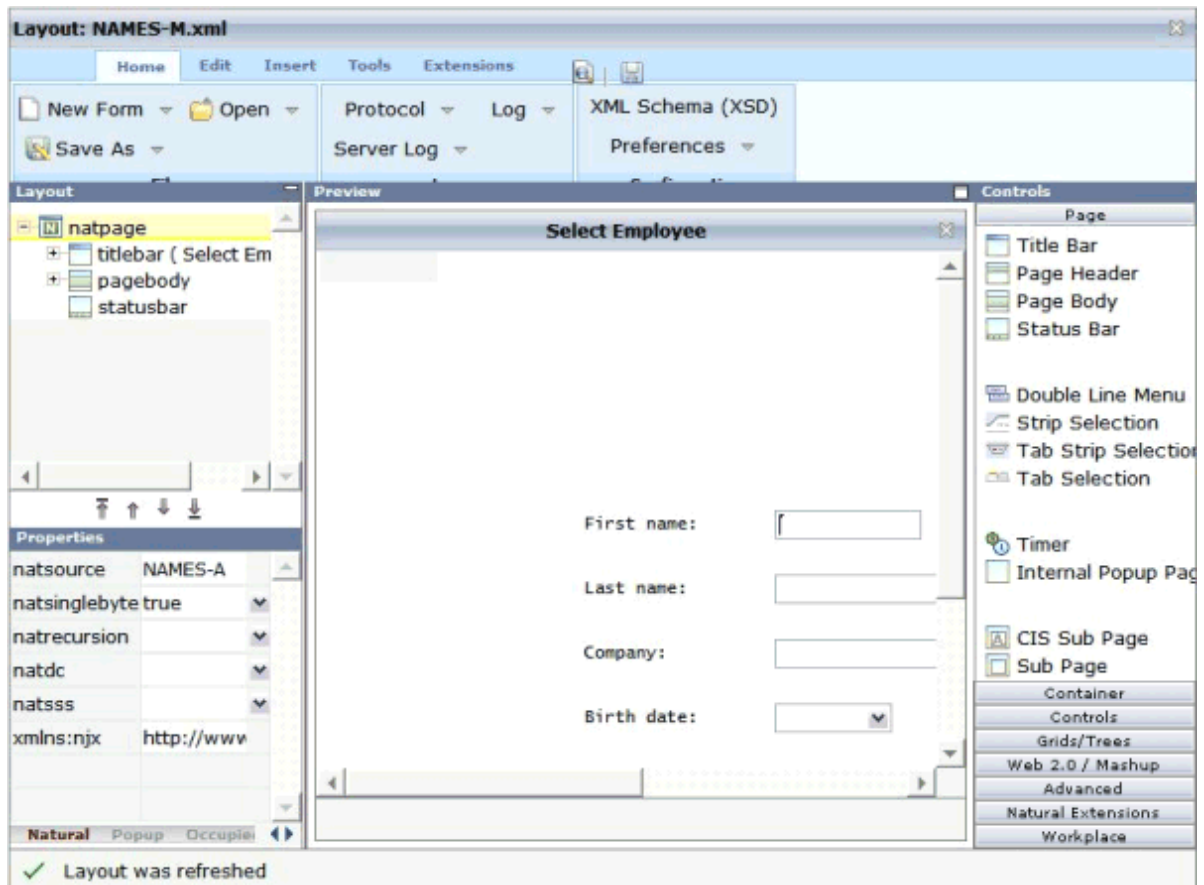
- 9 From the **Map Conversion** menu, choose **Preview in Browser** to display the resulting page layout in a separate browser window.

The image shows a software window titled "Select Employee". Inside the window, there are four text input fields arranged vertically, each preceded by a label: "First name:", "Last name:", "Company:", and "Birth date:". Below these fields, in the bottom-left corner of the window, is a legend with three entries: "F3 Exit", "F4 Delete", and "F5 Save". Each entry consists of a small square icon followed by the text. The window has a standard title bar with a close button in the top-right corner.

- 10 After having previewed the conversion result for one or several maps in your project, choose **Generate All Layouts** from the **Map Conversion** menu to generate page layouts for all map extract files contained in the selected folder.



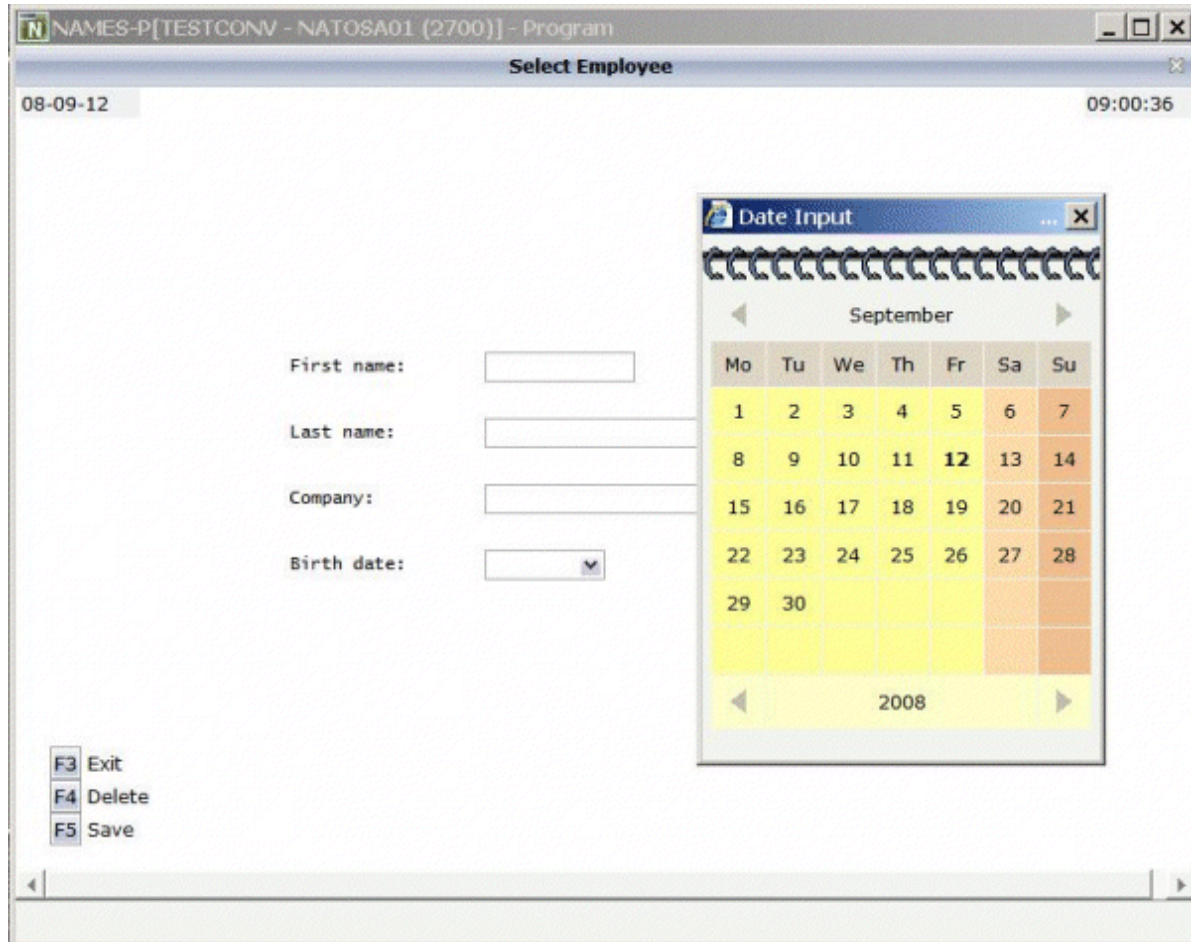
- 11 For now close the Map Converter and switch to the project *testconv* to continue working on the generated page layout.



- 12 You might wish to assign a different name for the adapter to be generated for the page, change other properties or modify the layout in any other way. Then save the layout and generate the adapter as usual.

When you import the adapter into your Natural library, you will notice that the parameter data area is the same as in the original map. This is the case even though the map uses system variables and variables with special characters. The necessary translation is done inside the generated adapter code and does not influence the application code.

- 13 Now create a main program for the adapter and run it in the browser.



You may have noticed the following effects of the applied conversion rules:

- The title in the first row of the map has been placed into the caption of the page and the asterisks have been stripped off. Your application will quite surely have a different layout of the map titles. The conversion rules can therefore be adapted to accommodate the needs of your application, and the rule that identifies the title and places it into the caption is just a simple application of customizing the conversion rules.
- The literals such as "F4 Delete" on the map have each been turned into a button control and a label. This is also due to a sample conversion rule contained in the default conversion rules.
- The date field has been converted to a field control with the data type "date". This enables the user to select the date with the **Date Input** dialog box.

The full concept of customizing the Map Converter is described in [Customizing the Conversion Process](#).

Using the Map Converter

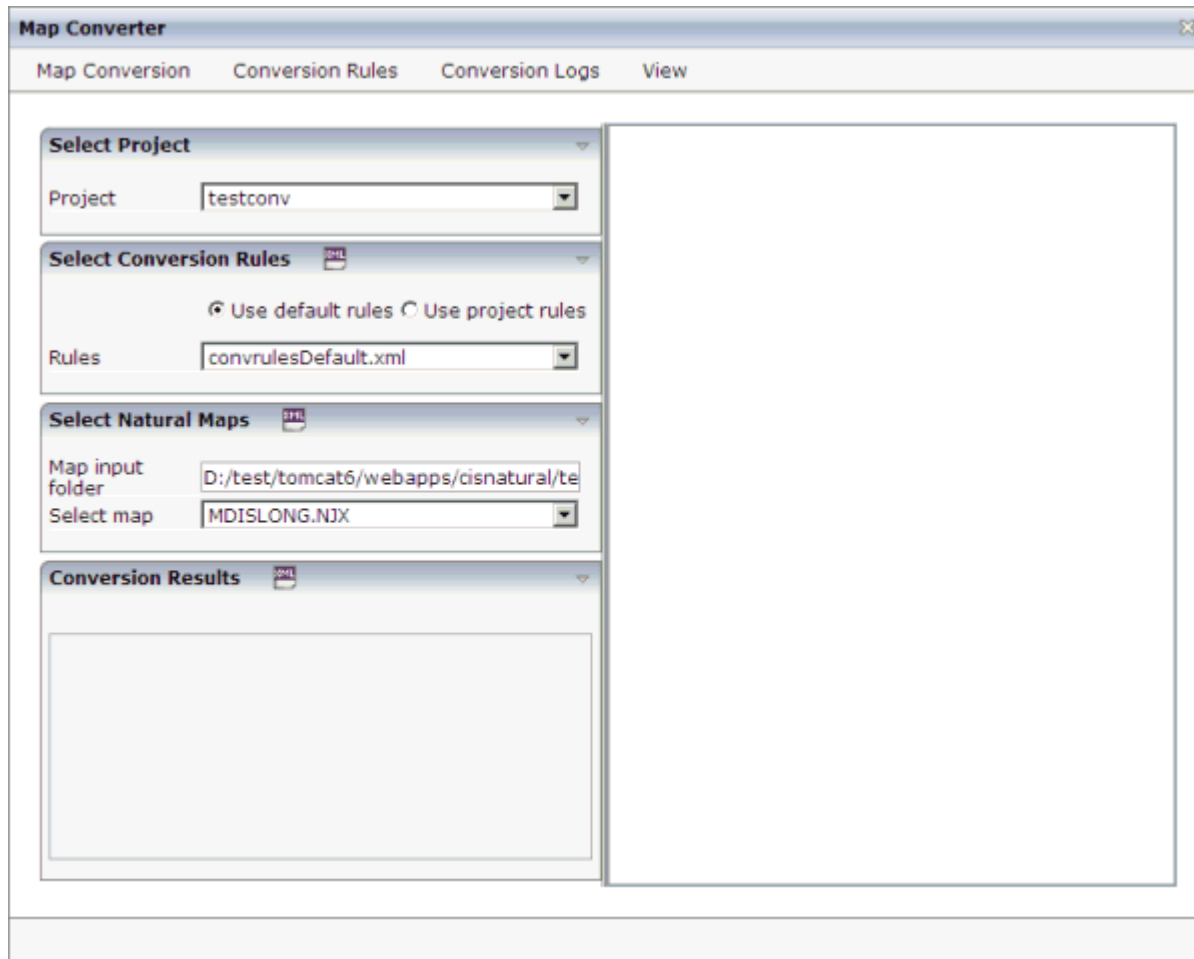
The Map Converter is used for mass generation of layouts.

In the *First Steps*, you have already learned how to use the Map Converter. The topics below provide detailed descriptions of the different options and menu commands that are available in the Map Converter:

- [Invoking the Map Converter](#)
- [Setting the Conversion Options](#)
- [Previewing/Generating a Single Layout](#)
- [Generating All Layouts](#)
- [Viewing the Conversion Results](#)
- [Refreshing the Display](#)

Invoking the Map Converter

When you invoke the Map Converter, the following dialog appears.



► To invoke the Map Converter

- In the **Natural Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Map Converter**.

Setting the Conversion Options

In order to start the generation, you have to select a project, a conversion rules file and the folder containing your map extract files. The following options are available for this purpose:

Project

This drop-down list box provides for selection all Application Designer projects that are currently defined.

Select the project in which you want to store the page layouts that are to be generated.

Use default rules

When this option button is selected, the default conversion rules and related templates are used. These rules are stored in the subdirectory *convrules* of the project directory *njxmapconverter*.

Use project rules

When this option button is selected, the project-specific conversion rules are used. These rules are contained in the subdirectory *convrules* of your project directory.

When your project does not yet have any project rules and you select this option button, the Conversion Tool is automatically shown in a dialog. You can then copy the default conversion rules and templates to the currently selected project. It is recommended that you copy all or part of the default rules and related templates into your project and adapt the copies to the requirements of your application. See [Using the Conversion Rules Tool](#) for further information.

You can also invoke the Conversion Tool manually. To do so, you choose **Copy Rules** from the **Conversion Rules** menu.

Rules

This drop-down list box provides for selection all available conversion rules files. When the **Use default rules** option button is selected, the default rules files are shown. When the **Use project rules** option button is selected, the rules files in the project directory are shown.

Select the conversion rules file that is to be used.

You can display the XML code of the selected conversion rules file in a dialog. To do so, you either choose the icon that is shown in the **Select Conversion Rules** header or you choose **Show Rules** from the **Conversion Rules** menu.

Map input folder

Specify the folder which contains the map extract files that are to be processed.

Select map

Optional. This drop-down list box provides for selection all map extract files that are stored in the currently selected map input folder.

For mass generation, it is not required that you select a map. However, you can select a map, for example, if you want preview the layout of the resulting Application Designer page as it would turn out using the selected conversion rules file.

You can display the XML code of the selected map extract file in a dialog. To do so, you either choose the icon that is shown in the **Select Natural Maps** header or you choose **Show Map** from the **Map Conversion** menu.

Previewing/Generating a Single Layout

When you choose one of the following commands from the **Map Conversion** menu, the currently selected conversion rules file and the currently selected map extract file are used for preview or generation of a single layout from a single map extract file:

Preview Page Layout

Shows a single page layout in the preview area of the Map Converter (on the right side).

Preview in Browser

Shows a preview of a single page layout in a separate browser window.

Generate Selected Layout

Generates a single page layout. The resulting file is stored in the currently selected project.

Generating All Layouts

When you choose the following command from the **Map Conversion** menu, the currently selected conversion rules file and all map extract files in the selected map input folder are used as input for the mass generation:

Generate All Layouts

Generates all page layouts (mass generation). The resulting files are stored in the currently selected project.

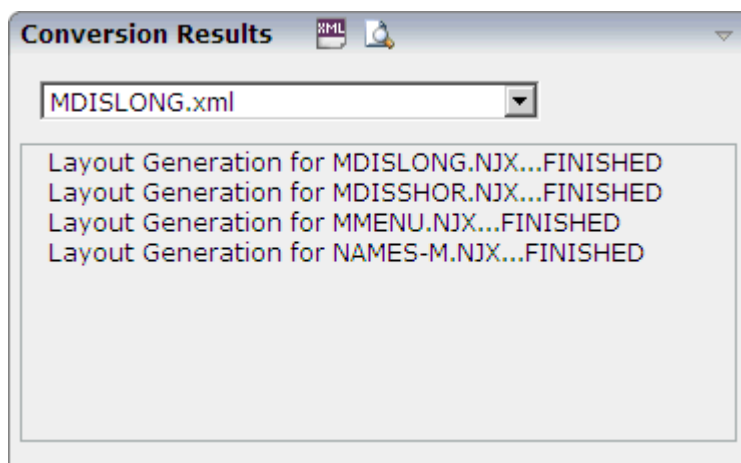
Viewing the Conversion Results

After a preview or generation, you can either choose the icon that is shown in the **Conversion Results** header or you choose the following command from the **Map Conversion** menu:

Show Layout XML Definition

Shows the XML layout definition for the page which was last generated or previewed in a dialog.

When the last generation was a mass generation, an additional drop-down list box is shown under **Conversion Results**. This drop-down list box provides for selection the names of all generated page layouts. When you choose the **Show Layout XML Definition** command (or the corresponding icon), the XML layout definition for the page which is currently selected in the drop-down list box is shown in a dialog.



After a mass generation, an additional icon for previewing a generated page layout is shown in the **Conversion Results** header. When you choose this icon, the layout for the page which is currently selected in the drop-down list box is shown in the preview area of the Map Converter (on the right side).

When you choose the **Show Logs** command from the **Conversion Logs** menu, the Conversion Logs tool is shown in a dialog. For further information on the options in this dialog, see [Using the Conversion Logs Tool](#).

Refreshing the Display

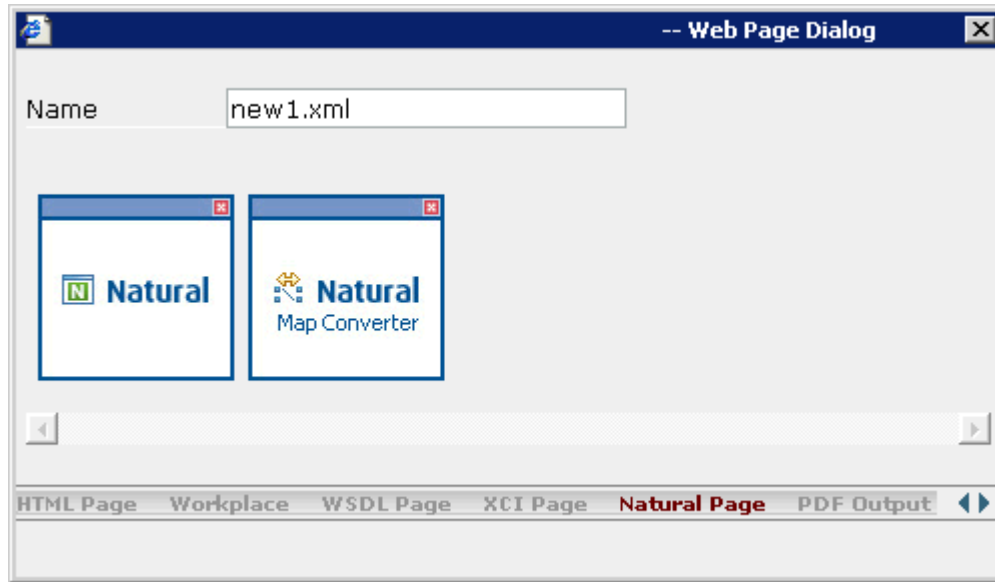
For example, when you have created a new project which is not yet visible in the Map Converter, you can choose the **Refresh** command from the **View** menu of the Map Converter. This reloads all projects, conversion rules and map extract files and resets the contents of the dialog.

Using the Editor Extension

An editor extension, the Map Conversion Assistant, is used to generate a single layout while designing a page in the Layout Painter. In this case, you fill an empty layout with the information from a map extract file.

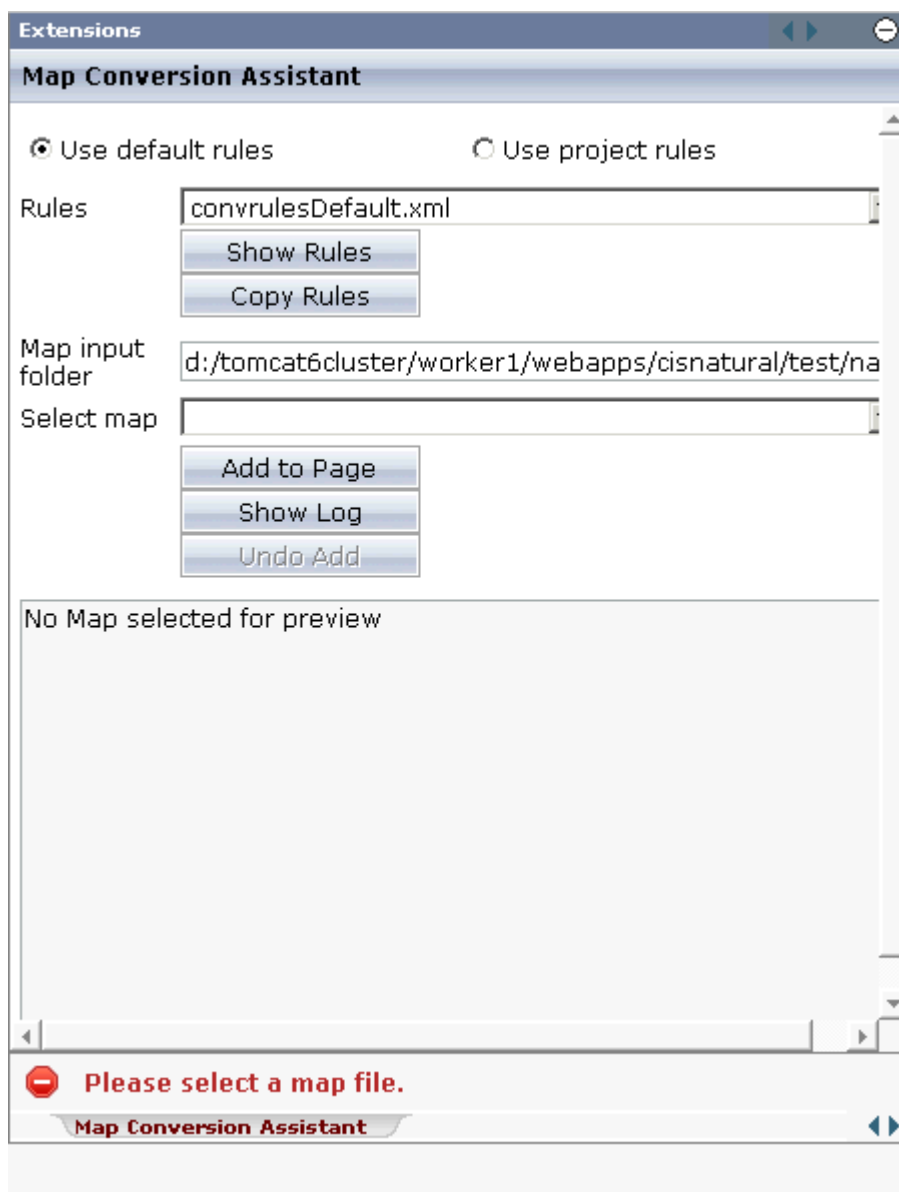
► To add a map to an empty layout using the editor extension

- 1 Create a new layout using the Natural Map Converter template.



- 2 From the **Extensions** tab of the Layout Painter, choose **Map Conversion Assistant**.

The following area is now shown in the Layout Painter.



- 3 Select either the **Use default rules** option button or the **Use project rules** option button. See [Setting the Conversion Options](#) for information on these option buttons.
- 4 Optional. When you choose the **Copy Rules** button, you can copy the default conversion rules and templates to the current project. In this case, the Conversion Rules tool is shown in a dialog. For further information on the options in this dialog, see [Using the Conversion Rules Tool](#).
- 5 From the **Rules** drop-down list box, select the conversion rules file that is to be used. The rules files that are provided for selection in this drop-down list box depend on the setting of the option buttons (either the default rules or the project rules are shown).
- 6 Optional. When you choose the **Show Rules** button, the XML code of the selected conversion rules file is shown in a dialog.

- 7 In the **Map input folder** text box, specify the folder which contains the map extract files.
- 8 From the **Select map** drop-down list box, select the map that is to be used.

The XML layout definition of the selected map is now shown at the bottom of the Map Conversion Assistant.

- 9 Choose the **Add to Page** button.

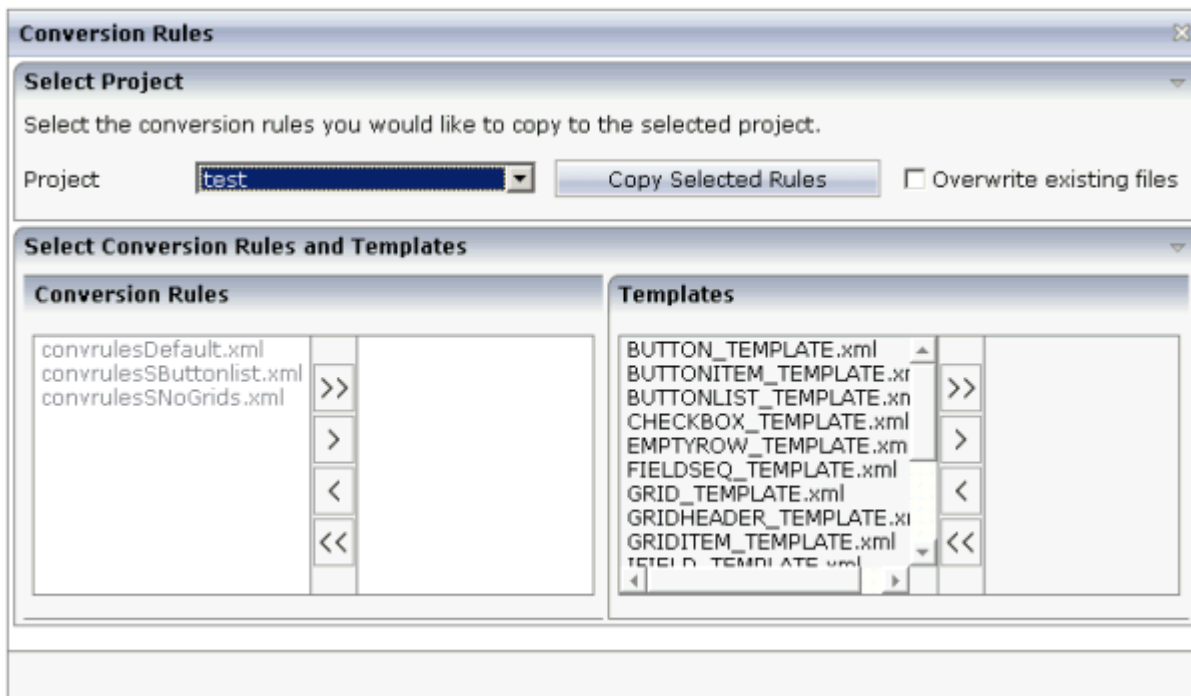
The map description is converted to the corresponding layout elements and these elements are added to the current layout, which is now shown in the preview area.

The **Add to Page** button is now dimmed. If you want to remove the elements you have added to the page, you can choose the **Undo Add** button.

- 10 Optional. When you choose the **Show Log** button, the Conversion Logs tool is shown in a dialog. For further information on the options in this dialog, see [Using the Conversion Logs Tool](#).
- 11 Modify the layout as usual.

Using the Conversion Rules Tool

Using this tool you can copy the default conversion rules and templates to a selected project for modification.



► **To invoke the Conversion Rules tool**

- In the **Natural Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Conversion Rules**.

Or:

When the **Map Converter** is currently shown, choose **Copy Rules** from the **Conversion Rules** menu.

Or:

When the **editor extension** is currently shown, choose the **Copy Rules** button.

► **To copy the conversion rules**

- 1 From the **Project** drop-down list box, select the project into which you want to copy the conversion rules.
- 2 In the **Conversion Rules** box, select the rules file(s) that you want to copy and choose the > button.

Or:

If you want to copy all files, choose the >> button.

The selected files are shown on the right side of the **Conversion Rules** box.

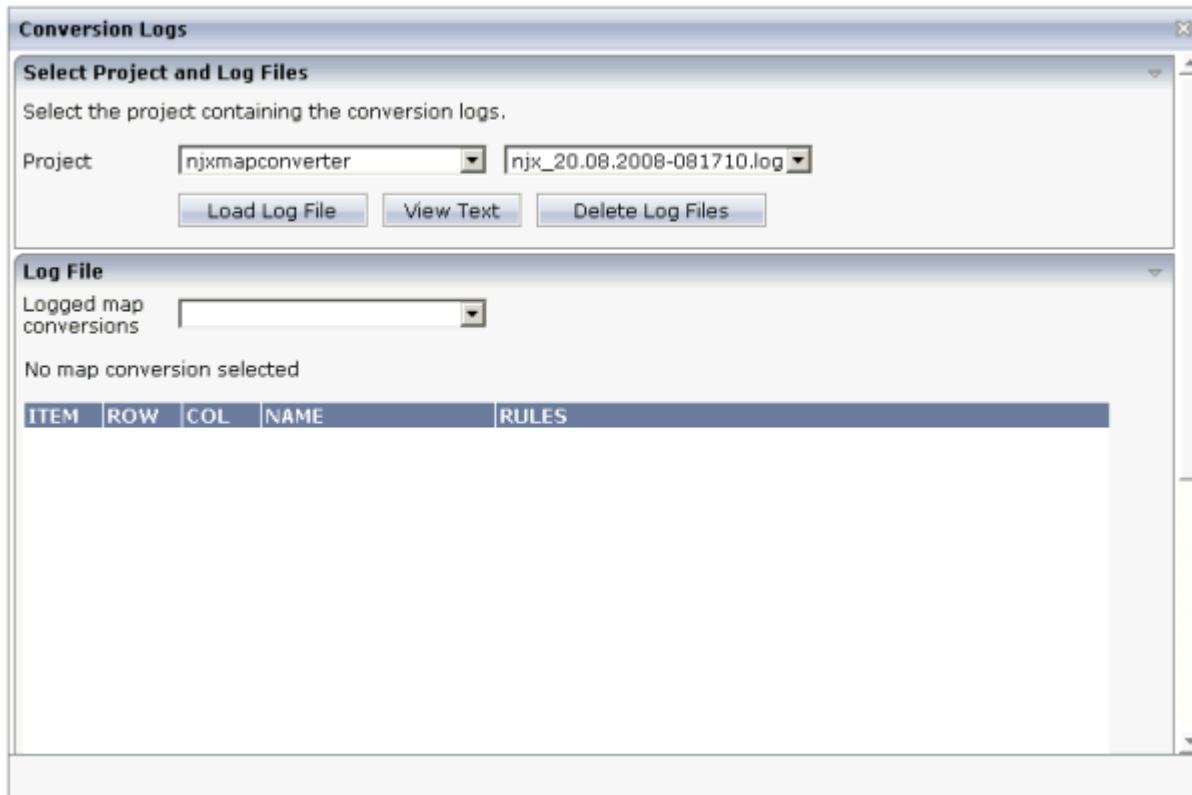
To deselect one or more files, you can use the < or << button.

For each selected rules file, the templates that are used in the rules file are automatically selected in the **Templates** box, so that always a consistent set of rules and templates is selected for copying.

- 3 Optional. If you want to overwrite any existing rules and templates files with the same names in the selected project, activate the **Overwrite existing files** check box.
- 4 Choose the **Copy Selected Rules** button to copy the rules and templates files to the selected project.

Using the Conversion Logs Tool

Using this tool you can view the log files that have been created during the conversion of Natural maps to Application Designer layouts. You can also delete these log files.



► To invoke the Conversion Logs tool

- In the **Natural Tools** node of the navigation frame (which is visible when the **Tools & Documentation** button has previously been chosen), choose **Conversion Logs**.

Or:

When the **Map Converter** is currently shown, choose **Show Log** from the **Conversion Logs** menu.

Or:

When the **editor extension** is currently shown, choose the **Show Log** button.

► **To view a log file**

- 1 From the **Project** drop-down list box, select the project for which you want to view a log file.
The log files contained in this project are shown in the drop-down list box to the right.
- 2 Select the log file that you want to view.
- 3 Choose the **Load Log File** button.
Log lines for the selected log file are now shown at the bottom of the tool. Each log file contains the conversion results of one or several maps. The log lines that are shown belong to an individual map; this is the map that is selected in the **Logged map conversions** drop-down list box.
- 4 Optional. Select a different map from the **Logged map conversions** drop-down list box.
The conversion result of the newly selected map is immediately shown at the bottom of the tool.
- 5 Optional. Choose the **View Text** button to display the content of the selected log file as a CSV file in a dialog. This shows the conversion results for all maps.

► **To delete log files**

- 1 Select the project for which you want to delete the log files.
- 2 Choose the **Delete Log Files** button.
A dialog appears asking to confirm the deletion.
- 3 Choose the **Yes** button to delete all log files in the selected project.

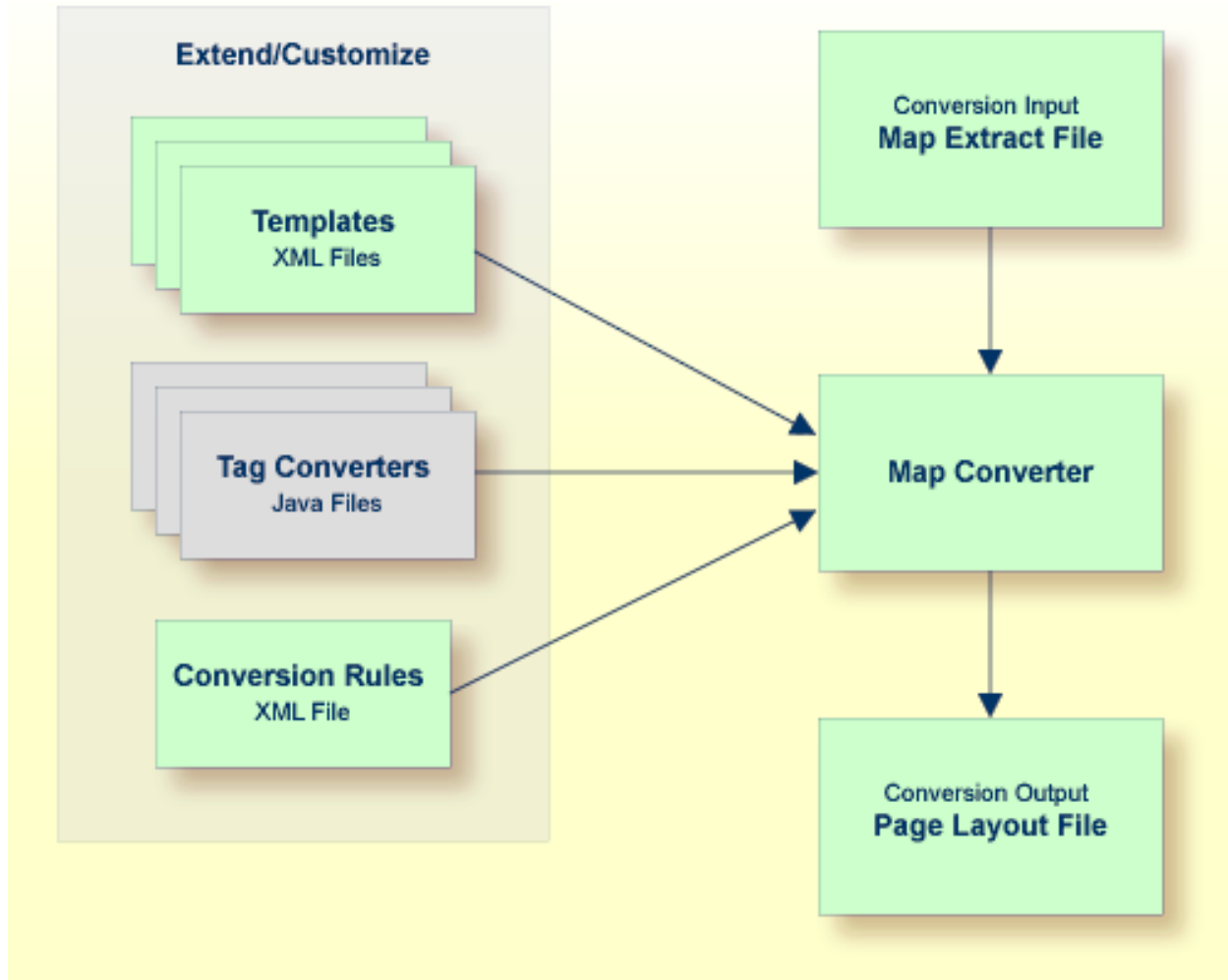
26

Customizing the Map Conversion Process

■ Map Converter Processing	150
■ Conversion Rules	154
■ Templates	164
■ Tag Converters	167

Map Converter Processing

The map conversion process reads a map extract file created by the Map Extractor or the `INPUT` Extractor and transforms it into a corresponding Application Designer page layout file. The conversion process is controlled by rules and templates.



The Map Converter ships with a default set of conversion rules and corresponding template files. This set allows for default map conversions without changing rules or templates. In most cases, you will add or modify some conversion rules and/or templates to customize the conversion according to the requirements of your application.

For advanced customizations, there is also the possibility to plug own Java-written conversion classes (the so-called “tag converters”) into the conversion processing. But you should only do this in very rare cases.

The following topics are covered below:

- [Processing of Rows and Columns](#)
- [Processing of Sequence and Grid Areas](#)
- [Summary: Processing Steps of the Map Converter](#)

Processing of Rows and Columns

By default, for each row and column in a map, a corresponding row and column is generated in the layout. By default, the Map Converter inserts the converted rows and columns at a defined position within a corresponding page template. Template and insert position can be defined by the user. Skipping or different handling of specific rows and columns can be defined via corresponding conversion rules.

The following sections describe the default processing for rows and columns in case no specific rules for different insert positions are specified:

- [Rows](#)
- [Columns](#)

Rows

For each row in a map, the Map Converter generates an ITR (independent table row) control with the default settings. For empty rows, an ITR control containing the control defined in the `EMPTYROW_TEMPLATE` is generated.

Columns

The fields and literals within a row are aligned to columns according to the following rules:

■ Column Start Position

If an absolute column start position is defined for a field or literal in the map, the corresponding control in the page layout is aligned so that it starts exactly with the specified column. This is done by inserting a HDIST (horizontal distance) control with a corresponding width as a filler.

■ Conversion Rules

If no absolute column start position is defined for a field or literal in the map, a HDIST control is not added as a filler by default. In this case, the field or literal is simply appended as the last subnode of the current ITR control. In many cases, this would result in a layout that requires additional manual adding of fillers. This is because appending two field controls without adding any HDIST control often does not look as intended. Therefore, the Map Converter includes default conversion rules for filler settings. You can modify the default conversion rules or add your own conversion rules to fine-tune this behavior. For more information, see [Conversion Rules](#).

■ Column Width

A character map has a fixed number of rows and columns. For the literal "ABCD", this means that it uses exactly 4 columns. Calculating the correct width and height of field on a web page is more complex. The width of "ABCD" will most likely be greater than the width of "IIII". Very short fields (with a length of one or two characters) should have a minimum width so that the content is fully visible. You can fine-tune the width by adapting the predefined conversion rule variable `$$widthfactor$$` or by adding your own conversion rules. For more information, see [Conversion Rules](#).

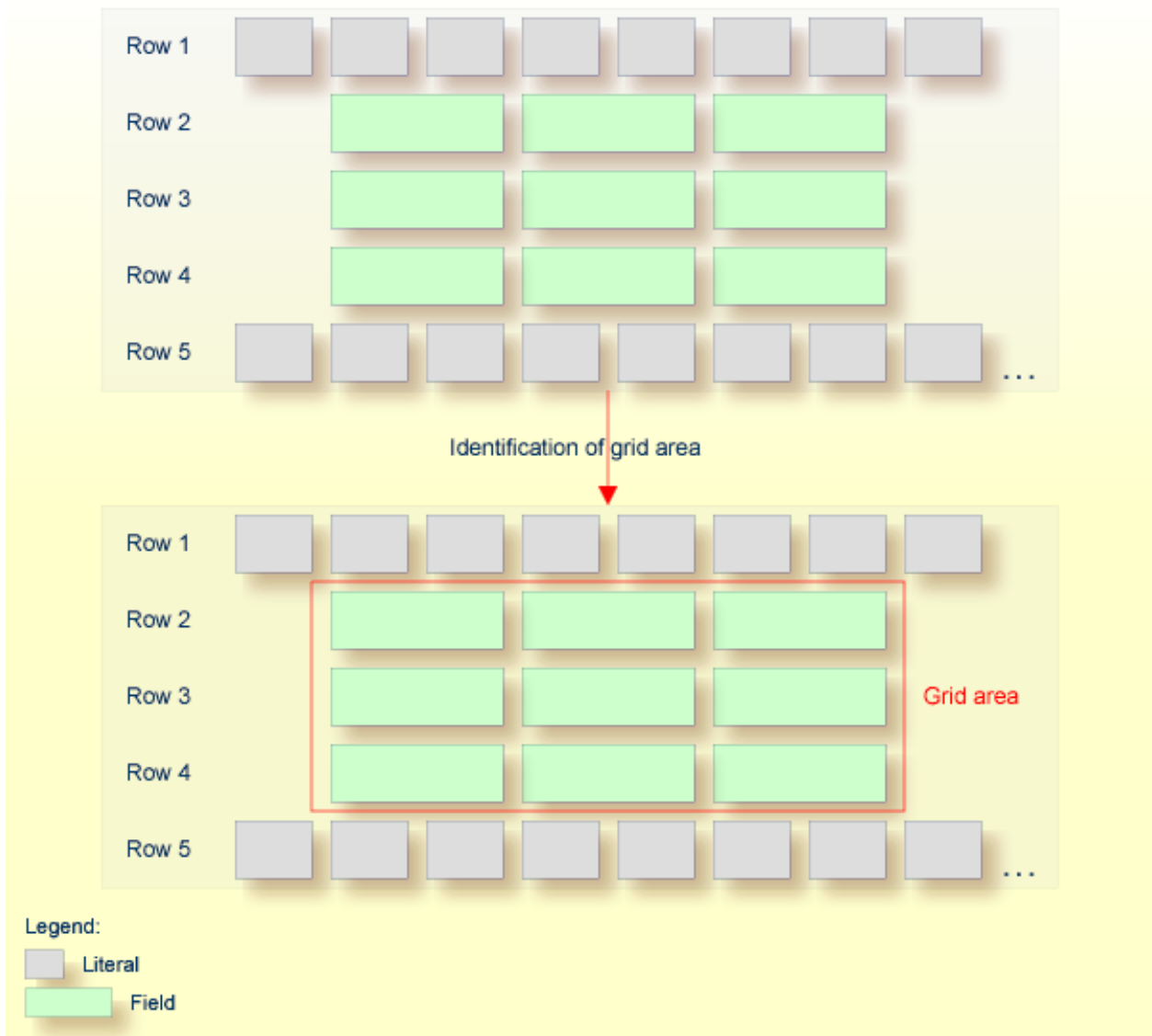
Processing of Sequence and Grid Areas

The map extract file also contains information about arrays. With Application Designer, arrays are usually rendered as grid controls. Application Designer provides a couple of grid controls:

- **TEXTGRID2** - a grid containing text.
- **TEXTGRIDSSS2** - a text grid with server-side scrolling.
- **ROWTABLEAREA2** - a grid containing other controls.
- **MGDGRID** - a managed grid.

The Map Converter tries to convert arrays into suitable grid controls. Before the real conversion of arrays to grid controls can be done, the Map Converter must first identify the sequence and grid areas on the map. During this process of area identification, the Map Converter groups literals and fields together into sequences and areas. Whether the corresponding fields or literals are actually converted into a grid depends on the conversion rules that are executed after this area identification step.

This process of area identification is simply a kind of marking. The corresponding sequence and area objects can be used as source in the conversion rules to define the actual controls.



Summary: Processing Steps of the Map Converter

The conversion is done in several steps:

1. The map extract file is loaded and the corresponding rows and columns are collected.
2. The sequence and grid areas are identified.
3. For each row, the list of items in this row is processed, according to the column order. An item can be one of the following: a simple literal, a field or an area. For each found item, the corresponding conversion rules are executed.

Conversion Rules

Different conversion projects have different requirements to the conversion process. The Map Converter is driven by conversion rules and thus allows for flexible control of the conversion process. Conversion rules define how source items (items from a given map extract file) are mapped to target items (items in the page layout to be created) and under which conditions a certain source item shall be converted to a certain target item. The Map Converter is delivered with a default set of conversion rules contained in the file *convrulesDefault.xml* in the subdirectory *convrules* in the Application Designer project *njxmapconverter*. A more application-specific conversion can be achieved by copying and modifying the default set of rules or by adding own rules.

Each set of conversion rules is defined in an XML file according to the XML schema *convrules.xsd* in the subdirectory *convrules* in the Application Designer project *njxmapconverter*. Each individual conversion rule consists of a name, a description, a source and a target. The source identifies an element in the map extract file. The target identifies controls and attributes to be generated in the page layout.

The conversion rules make often use of regular expressions and so-called capture groups. For more information about regular expressions, see for instance the web site <http://www.regular-expressions.info>.

The following topics are covered below:

- [Conversion Rules Examples](#)
- [Default Conversion Rules File](#)
- [Conversion Rules that Often Need to be Adapted](#)
- [Writing Your Own Conversion Rules](#)

Conversion Rules Examples

The following examples are provided:

- [Example 1](#)
- [Example 2](#)

■ Example 3

Example 1

The following example rule (contained in the default conversion rules file) defines that fields in the map extract file with the qualification AD=0 shall be converted to field controls with the property `displayonly="true"`.

```
<convrule rulename="Ofield_rule">
  <description>Defines the control template to be used for input fields
  which are specified as output only.</description>
  <source>
    <sourceitem>ifField</sourceitem>
    <sourcecond>
      <condattr>//ifAD</condattr>
      <condvalue>.*0.*</condvalue>
    </sourcecond>
  </source>
  <target>
    <targetitem>$OFIELD_TEMPLATE</targetitem>
  </target>
</convrule>
```

The source element specifies that this rule applies to fields (element `ifField`) that have an AD parameter (element `ifAD`) that contains a letter "O" (matching the regular expression `.*0.*`). The target element specifies that these fields are to be converted to whatever is contained in the template file `OFIELD_TEMPLATE.xml`. This template file must be contained in the same directory as the conversion rules file.

The template file contains the detailed specification of the field to be generated. The file `OFIELD_TEMPLATE.xml` delivered with the map converter contains, for instance, the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<field valueprop="$$" width="$$" noborder="true" displayonly="true"/>
```

That is, the resulting field is generated without a border (`noborder="true"`) and as a display-only field (`displayonly="true"`). The `valueprop` and `width` to be assigned (`$$`) are not determined by this rule, but are left under the control of other rules.

Example 2

The following example rule (contained in the default conversion rules file) defines that for all fields that are defined with the format A_n in the map extract file, an attribute `datatype="string n"` shall be added to the element that is generated into the page layout.

```
<convrule rulename="AfixType_rule">
  <description>All Natural "An" dfFields are converted to the
  Application Designer datatype "string n". Example: "A10" is
  converted to "string n".</description>
  <source>
    <sourceitem>dfField</sourceitem>
    <selection>
      <selectattr>dfFormat</selectattr>
      <selectval>A([0-9]+)</selectval>
    </selection>
  </source>
  <target>
    <targetitem>$$</targetitem>
    <targetattr>
      <attrname>datatype</attrname>
      <attrvalue>string $1</attrvalue>
    </targetattr>
  </target>
</convrule>
```

The source element specifies that this rule applies to fields that have in the field definition (element `dfField`) a format (element `dfFormat`) of A_n (matching the regular expression `A([0-9]+)`). The target element specifies that for whatever element is generated into the page layout for this kind of fields, an attribute `datatype="string $1"` shall be added. In terms of regular expressions, `$1` refers to the contents of the first “capture group” of the regular expression `A([0-9]+)`. In case of a format `A20`, `$1` will evaluate to `20` and thus an attribute `datatype="string 20"` will be generated.

The control to be generated into the page layout (`<targetitem>$$</targetitem>`) is not determined by this rule, but is left under the control of other rules.

Summary: The combination of the two rules in example 1 and 2 makes sure that output fields, for example, of format `A20` are converted to field controls with `displayonly="true"` and `datatype="string 20"`.

Example 3

The following more advanced rule was created for the use of a specific conversion project. The following task had to be achieved: A literal of the format "F10 Change" shall be converted to a button that is named "F10", is labeled "Change" and raises an event named "PF10". With the explanations from the examples above, the rule should be nearly self-explanatory.

Note that according to the rules of regular expressions, the variable \$1 refers to the string matched by the expression part in the first pair of parentheses (the first “capture group”), that is for instance "F10", and the variable \$3 refers to the string matched by the expression part in the third pair of parentheses (the third “capture group”), that is for instance "Change".

```
<convrule rulename="Function_rule" lone="true">
<description>Generates a button from specific literals.</description>
  <source>
    <sourceitem>ltLiteral</sourceitem>
    <selection>
      <selectattr>ltName</selectattr>
      <selectval>(F[0-9]+)(\p{Space})(.*)</selectval>
    </selection>
  </source>
  <target>
    <targetitem>$BUTTON_TEMPLATE</targetitem>
    <targetattr>
      <attrname>name</attrname>
      <attrvalue>$1</attrvalue>
    </targetattr>
    <targetattr>
      <attrname>method</attrname>
      <attrvalue>P$1</attrvalue>
    </targetattr>
  </target>
  <target>
    <targetitem>hdist</targetitem>
    <targetattr>
      <attrname>width</attrname>
      <attrvalue>4</attrvalue>
    </targetattr>
  </target>
  <target>
    <targetitem>label</targetitem>
    <targetattr>
      <attrname>name</attrname>
      <attrvalue>$3</attrvalue>
    </targetattr>
  </target>
</convrule>
```

Default Conversion Rules File

The Map Converter is delivered with a default set of conversion rules contained in the file *convrulesDefault.xml* in the subdirectory *convrules* in the Application Designer project *njxmapconverter*. A more application-specific conversion can be achieved by copying and modifying the default set of rules or by adding own rules.

The following topics are covered below:

- [Root Rule](#)
- [Data Type Conversion Rules](#)
- [Other Default Conversion Rules](#)

Root Rule

Like every conversion rules file, the file contains exactly one "Root_rule". The root rule specifies the template file to be used for the overall page layout. In this template file, the application-specific page layout can be defined, using company logos, colors, fonts, etc. The root rule must always have "map" as the source item and must refer to some variable defined in the page template file as the target item. The place of that variable specifies where in the page template the converted map items are placed. See for instance the root rule from the default conversion rules:

```
<convrule rulename="Root_rule">
  <description>Exactly one rule with the sourceitem "map" is required.
  This rule must define the natpage template and insert position of
  the conversion result.</description>
  <source>
    <sourceitem>map</sourceitem>
  </source>
  <target>
    <targetitem>$NATPAGE_TEMPLATE.$MAPROOT</targetitem>
  </target>
</convrule>
```

The rule refers to a page layout template *NATPAGE_TEMPLATE.xml* and refers to a variable defined in that template where the converted map elements shall be placed. Here is the corresponding content of the page layout template *NATPAGE_TEMPLATE.xml*:

```
<?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>
```

This template specifies the following:

- The overall page layout shall consist of the elements `titlebar`, `pagebody` and `statusbar`.
- The converted map elements shall be placed into the `pagebody`.
- The name of the Natural adapter to be generated from that page layout shall be determined by a rule (`natsource="$$NATSOURCE$$"`). There must be a corresponding rule that yields a value for the variable `$$NATSOURCE$$`, for instance derived from the map name. We shall see later how to define such a rule.
- All strings in the page layout shall be mapped to Natural variables of type A in the adapter interface (`natsinglebyte="true"`).
- The text displayed in the title bar shall be determined by a rule (`name="$$TITLEVAR$$"`). There must be a corresponding rule that yields a value for the variable `$$TITLEVAR$$`, for instance derived from a literal in the first row in the map. We shall see later how to define such a rule.

Data Type Conversion Rules

The default conversion rules file contains a set of rules that control the conversion of data types: from Natural data types in the map to corresponding Application Designer data types in the page layout. An example was given above in [Example 2](#). Usually, these rules need not be adapted. They have been chosen in such a way that the process of extracting maps, converting them to layouts and generating Natural adapters for these usually yields the same data types in the adapter interface as in the map interface.

Other Default Conversion Rules

Other default conversion rules define a default mapping for literals, modifiable fields, output fields, modifiable grids, output grids, system variables and fields with special characters like "#" in their names. These rules need only be adapted in special cases.

Conversion Rules that Often Need to be Adapted

Some conversion rules need to be adapted in nearly all conversion projects. These rules are contained in the section "APPLICATION SPECIFIC RULES" in the default conversion rules file.

The following topics are covered below:

- [Naming of Adapters](#)

- [Setting the Title of a Map](#)

Naming of Adapters

Each application has a different naming convention for Natural objects. There is a rule (it is named "Natsource_rule" in the default conversion rules file) that controls how adapter names are derived from map names. The rule replaces the first letter "M" in the map name with an "A" and places the resulting string into the variable `NATSOURCE`. Remember that in the default page template, the `natsource` property of `NATPAGE` (which defines the adapter name to generated) is preset with the variable reference `$$NATSOURCE$$`. Thus, a map with the name `TESTM1` results in an adapter named `TESTA1`. Other naming conventions for maps will require a more sophisticated adapter naming rule.

Setting the Title of a Map

Each application has a different way of showing titles in a map. Often, the title string shall be placed into the title bar of the resulting page layout during conversion. There is a rule (in the default conversion rules file, it is named "Titlevar_rule") that controls how the title string in a map is recognized. The rule searches in the first row of a map for a literal enclosed in "****" and places the resulting string into the variable `TITLEVAR`. Remember that in the default page template, the `name` property of the `titlebar` element (which defines the string to be shown in the title bar) is preset with the variable reference `$$TITLEBAR$$`. So this rule takes care that the found literal is placed into the `titlebar` element of the page. Other conventions for map titles will require a more sophisticated rule.

Writing Your Own Conversion Rules

When writing your own conversion rules, you can use the default rules as examples. In order to write rules from scratch, you need to know the elements of the map that can be referred to as source items and the full syntax of the rule definition.

- The XML schema of the map extract files is contained in the file *naturalmap.xsd* in the subdirectory *convrules* in the Application Designer project *njxmapconverter*.
- As described in [Processing of Sequence and Grid Areas](#), one step in the map conversion is the detection of sequence and grid areas in the map. Conversion rules can also refer to the detected sequence and grid areas. The XML schema of the map extract files after the detection of sequence and grid areas is described in the extended XML schema *naturalmapxml_extended.xsd* in the same directory.
- The syntax of the conversion rules is described by the XML schema *convrules.xsd* in the same directory.

The basic structure of a conversion rule is as follows:

```
<convrule rulename="...">
  <description>...</description>
  <source>...</source>
  <target>...</target>
  <target>...</target>
  ...
</convrule>
```

This means, a conversion rule consists of one `source` element and (optionally) one or several `target` elements. The `source` element identifies an item from the map. The `target` elements specify the conversion output. If no `target` elements are specified, nothing is generated from the identified `source` element.

The basic structure of a `source` element is as follows (example):

```
<source>
  <sourceitem>ltLiteral</sourceitem>
  <selection>
    <selectattr>ltName</selectattr>
    <selectval>\*\*\*(.*)\*\*\*</selectval>
  </selection>
  <sourcecond>
    <condattr>ltRow</condattr>
    <condvalue>1</condvalue>
  </sourcecond>
</source>
```

The `sourceitem` element refers to a specific kind of item on a map, such as a literal (`ltLiteral`), a defined field (`dfField`), an input field (`ifField`) or the identifier of the map (`identity`). The elements that can be used here are specified by the XML schema that describes the map extract after the detection of sequence and grid areas (*naturalmapxml_extended.xsd*). Therefore, the elements `sequenceArea` and `gridArea`, which are only known after this processing, can also be used here.

The `selectattr` and `selectval` elements are used to match an element of a specific kind by its attribute values. The `selectval` element uses regular expressions to perform a match. Capturing groups such as `(.*)` can be used here, so that the target part of the conversion rule can later refer to parts of the matched value.

Finally, there can be zero, one or several `sourcecond` elements, which allow to define further to which map items the rule applies. If several `sourcecond` elements are specified, the rule is triggered only if all conditions match (logical AND).

The basic structure of a `target` element is as follows:

```
<target>
  <targetitem>...</targetitem>
  <targetattr>
    <attrname>...</attrname>
    <attrvalue>...</attrvalue>
  </targetattr>
  <targetattr>
    ...
  </targetattr>
  ...
</target>
```

In detail, there are several different options to specify a target item:

- Specify the root element name of an Application Designer control, along with its attributes and attribute values. The attribute value can be a constant, a variable or a reference to a capturing group from a regular expression in a `sourcecond` element of the same rule. In this case, the corresponding control is generated during conversion.

```
<target>
  <targetitem>label</targetitem>
  <targetattr>
    <attrname>height</attrname>
    <attrvalue>10</attrvalue>
  </targetattr>
  <targetattr>
    <attrname>width</attrname>
    <attrvalue>$$width$$</attrvalue>
  </targetattr>
  <targetattr>
    <attrname>name</attrname>
    <attrvalue>$1</attrvalue>
  </targetattr>
</target>
```

- Specify the name of a variable that is defined in the conversion rules file in a `convvariable` element.

```
<target>
  <targetitem>$$name$$</targetitem>
</target>
```


- Refer to the name of a template file, optionally along with attribute names and values. In this case, whatever is contained in the template file will be generated. Attribute definitions in the template file are replaced.

```
<target>
  <targetitem>$BUTTON_TEMPLATE</targetitem>
  <targetattr>
    <attrname>name</attrname>
    <attrvalue>$1</attrvalue>
  </targetattr>
  <targetattr>
    <attrname>method</attrname>
    <attrvalue>P$1</attrvalue>
  </targetattr>
</target>
```

- Refer to the name of a template variable and the name of a template file, separated by a dot. In this case, the template variable is replaced with whatever is contained in the template file.

```
<target>
  <targetitem>$GRIDITEM.$GRIDITEM_TEMPLATE</targetitem>
</target>
```

- Only in the root rule: Specify the name of a template file and the name of a template variable that is contained in this file, separated by a dot. In this case, the template variable is replaced with the entire result of the map conversion.

```
<target>
  <targetitem>$NATPAGE_TEMPLATE.$MAPROOT</targetitem>
</target>
```

- Specify "\$\$" as the target item. This is useful when writing a more general rule that is to apply after another more specific rule has already created a target item. The attributes specified along with the target item "\$\$" are applied to the already created target item, whatever this target item was.

```
<target>
  <targetitem>$$</targetitem>
  <targetattr>
    <attrname>datatype</attrname>
    <attrvalue>xs:double</attrvalue>
  </targetattr>
</target>
```

- Specify "\$." as the target item. This refers to the template that is currently being processed. The attributes specified along with the target item "\$." are applied to the current template.

```
<target>
  <targetitem>$.</targetitem>
  <targetattr>
    <attrname>$$NATSOURCE$$</attrname>
    <attrvalue>$1-A</attrvalue>
  </targetattr>
</target>
```

Templates

The Map Converter assembles page layouts from templates. Which templates are used, how they are assembled and how variables in templates are filled is controlled by the conversion rules.

A template file describes the general layout of an entire Application Designer page layout or of an individual Application Designer control. A template can contain variables and references to other templates. During conversion, the Map Converter resolves the structure of the templates and fills the variables with specific values, depending on the contents of the map.

A template file can describe a simple control such as a FIELD control or a more complex control such as a TEXTGRIDSSS2 control. For the same control, multiple templates may exist. For example, an *ofield_TEMPLATE* and an *ifield_TEMPLATE* may both be templates for the FIELD control. The *ofield_TEMPLATE* would be used for output fields, the *ifield_TEMPLATE* for modifiable fields. Which template is used for which subset of fields of the map is specified in the conversion rules.

Template files are well-formed XML files which contain control definitions. They are placed in the folder *convrules* of your Application Designer project directory. The file name must end with "_TEMPLATE.xml". The Map Converter ships with a set of default template files.

The following topics are covered below:

- [Variables in Templates](#)
- [Templates in Templates](#)

■ Editing Templates

Variables in Templates

As already seen in the examples above, templates can contain variables. Variables can be freely defined by the user. Example:

```
<?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>
```

■ Variables as placeholders for the property values of controls

An example is the variable `$$TITLEVAR$$` in the template above. If a template contains a variable such as `name="$$TITLEVAR$$"`, there must be a corresponding rule that yields a value for the variable `$$TITLEVAR$$`. The Map Converter replaces the variable with this value.

The built-in variable `$$` has a specific meaning. If it occurs as a property value, there is no specific rule needed to produce the value. Instead, the Map Converter receives the value from a so-called tag converter. Tag converters are Java classes that are delivered with the Map Converter. Exchanging or writing your own tag converters is an advanced way of extending the Map Converter and is usually not required. See [Tag Converters](#) for further information.

■ Variables as placeholders for controls and containers

An example is the variable `MAPROOT` in the template above. Such a variable is defined by inserting an `NJX:NJXVARIABLE` control (from the controls palette of the Layout Painter) into a template. As long as the XML of the template is well-formed, an `NJX:NJXVARIABLE` control can be inserted at any place in the template. Conversion rules refer to this variable as `$MAPROOT`. Notice that the value in the `name` property of an `NJX:NJXVARIABLE` control does not start with `$`. Instead, the `NJX:NJXVARIABLE` control itself defines that it is a variable. The `NJX:NJXVARIABLE` control is a special control in the **Natural Extensions** section of the Layout Painter's controls palette.

Templates in Templates

Templates can refer to other templates. This can be done via adding variables. The variable can serve as a placeholder for another template. The template name is defined via a corresponding rule.

Example (*GRID_TEMPLATE.xml*):

```
<?xml version="1.0" encoding="UTF-8"?>
<rowtablearea2 withborder="false" griddataprop="$$gridname$$" rowcount="$$" >
  <tr>
    <hdist></hdist>
    <njx:njxvariable name="GRIDHEADER" />
  </tr>
  <repeat>
    <tr>
      <hdist></hdist>
      <njx:njxvariable name="GRIDITEM" />
    </tr>
  </repeat>
</rowtablearea2>
```

This means: A conversion rule like the following maps a grid area detected in the map to a ROWTABLEAREA2 control and formats the header and rows as specified in the templates *GRIDHEADER_TEMPLATE.xml* and *GRIDITEM_TEMPLATE.xml*.

```
<convrule rulename="Griditem_rule">
  <description>Mapping rule for the items of grid.</description>
  <source>
    <sourceitem>gridArea//ifField</sourceitem>
  </source>
  <target>
    <targetitem>$$GRIDITEM.$$GRIDITEM_TEMPLATE</targetitem>
  </target>
  <target>
    <targetitem>$$GRIDHEADER.$$GRIDHEADER_TEMPLATE</targetitem>
  </target>
</convrule>
```

Editing Templates

Only NATPAGE templates (like the default NATPAGE template *NATPAGE_TEMPLATE.xml*) can be edited with the Layout Painter. Templates for individual controls must currently be edited using a text editor.

Tag Converters

A template must be a valid XML document. The root element must correspond to the root element of a valid Application Designer control. Templates can contain variables. A special variable is the variable `$$`.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<button name="$$" method="$$"></button>
```

Each template is processed by a so-called tag converter. Tag converters are in charge of resolving the variable `$$`. A tag converter is a Java class that must support a specific interface and be available in the class path of the Map Converter. Which tag converter is used depends on the root element of the template.

In the above example, the root element is the `BUTTON` control. The following rule applies:

- If a Java class with the name `com.softwareag.natural.mapconverter.converters.BUTTONConverter` is found in the Java class path, this Java class is used as the tag converter.
- Otherwise, the class `com.softwareag.natural.mapconverter.converters.DEFAULTConverter` is used as the tag converter.

In the above example, the Map Converter tries to find the class `BUTTONConverter` first. Since a specific tag converter for the `BUTTON` control is not delivered with the Map Converter, the class `DEFAULTConverter` is used as the tag converter.

In order to supply a custom tag converter for the `BUTTON` control, for instance, you would have to create a Java class `BUTTONConverter` that belongs to the package `com.softwareag.natural.mapconverter.converters` and make it available in the Java class path of the Map Converter.

Detailed information on how to write your own tag converters is provided in the Application Designer development workplace as Javadoc; see **Map Converter Extension API** in the **Natural Tools** node of the navigation frame (under **Tools & Documentation**).

27

Code Conversion

■ General Information	170
■ Generating Adapters	170
■ Structure of a Map-Based Application	170
■ Structure of a Natural for Ajax Application	171
■ Tasks of the Code Conversion	172
■ DEFINE DATA Statement	172
■ INPUT Statement	173
■ REINPUT Statement	174
■ PF-Key Event Handling	176
■ SET KEY Statement	177
■ Processing Rules	181
■ System Variables	181
■ Variable Names Containing Special Characters	182

General Information

After the **Map Converter** has been used to create page layouts from map extract files, the last step in the conversion process is adapting the application code to the new user interface. This step can either be performed manually or, with Natural Engineer, partly automatically. In the following, the manual code conversion is described.

Generating Adapters

First of all, it is necessary to generate HTML code and Natural adapters from the page layouts that have been created by the Map Converter. This is the same procedure as with page layouts that have been created manually with the Layout Painter. Then, the adapters are imported into the Natural development environment.

Structure of a Map-Based Application

In this context, we need not consider the application code as a whole, but only the layer that handles the user interface. Often, the user interface handling part of a map-based application is structured in the following way:

- DEFINE DATA
- Initialization
- REPEAT
 - INPUT [USING MAP *map-name*]
 - Includes client-side validations (processing rules)
 - Server-side validations
 - REINPUT or ESCAPE TOP
- DECIDE ON *PF-KEY
 - Function key handler 1
 - Processing
 - REINPUT or ESCAPE TOP
 - Function key handler 2
 - Processing
 - REINPUT or ESCAPE TOP

- Function key handler *n*
 - Processing
 - ESCAPE BOTTOM
 - ...
- END-DECIDE
- END-REPEAT
- Cleanup
- END

In practice,

- the REPEAT loop might or might not be there, and
- there might not be a clean DECIDE structure for the function key handlers. Instead, checks for the pressed function key might be spread all over the code.

However, accepting these differences, the above structure should match a large number of applications.

Structure of a Natural for Ajax Application

The corresponding part of a Natural for Ajax application looks as follows:

- DEFINE DATA
- Initialization
- REPEAT
 - PROCESS PAGE USING *adapter-name*
 - Includes client-side validations
 - Server-side validations
 - PROCESS PAGE UPDATE FULL
- DECIDE ON *PAGE-EVENT
 - Event handler 1
 - Processing
 - PROCESS PAGE UPDATE FULL **or** ESCAPE TOP
 - Event handler 2
 - Processing
 - PROCESS PAGE UPDATE FULL **or** ESCAPE TOP

- Event handler *n*
 - Processing
 - ESCAPE BOTTOM
 - ...
- END-DECIDE
- END-REPEAT
- Cleanup
- END

Tasks of the Code Conversion

The code conversion should achieve the following:

- It should be minimal invasive.
- It should not duplicate business code.
- The converted application should be able to run not only with the new user interface, but also in a terminal session, in a Natural Web I/O Interface session and in batch, if it did so before the code conversion.

In detail, the code conversion needs to deal with the statements and constructs mentioned below.

DEFINE DATA Statement

The `DEFINE DATA` statement must be extended because the data structures exchanged between a program and map are not fully identical to those exchanged between a program and the corresponding adapter.

The default conversion rules delivered with the Map Converter perform a data type mapping that tries to ensure that the data elements in the map interface are mapped to data elements of the same type and name in the adapter interface.

The Application Designer controls are usually not only bound to business data elements, but also to additional control fields. Which control fields these are depends on the way in which the elements of a map are mapped to Application Designer controls by the Map Converter rules. For instance, a `statusprop` can be assigned to a field, which results in an additional parameter in the parameter data area of the adapter. An array on a map can have been converted to a grid control with server-side scrolling. In this case, the additional data structures needed to control server-side scrolling need to be added to the `DEFINE DATA` statement.

statusprop

The `statusprop` is needed to control the error status or focus of a **FIELD** control dynamically (see [example 3](#) for the `REINPUT` statement below where it is used to replace the `MARK *field-name` clause). The default conversion rules contain a rule that creates a `statusprop` property for each map field that is controlled by a control variable. The adapter generator creates from this property a corresponding status variable and a comment line that identifies the status variable as belonging to the field.

Example

The parameter data area of the map contains:

```
01 LIB-NAME (A8)
01 LIB-NAME-CV (C)
```

The parameter data area of the adapter will then contain:

```
* statusprop= STATUS_LIB-NAME-CV
01 LIB-NAME (A8)
01 STATUS_LIB-NAME-CV (A) DYNAMIC
```

The variable `STATUS_LIB-NAME-CV` is not yet known to the main program and must be defined there.

INPUT Statement

The replacement for the `INPUT` statement is the `PROCESS PAGE` statement. In its simplest form, the `INPUT` statement just references the map. In this case, it is just replaced by a `PROCESS PAGE` statement with the corresponding adapter.

Example 1

Main program before conversion:

```
INPUT USING MAP 'MMENU'
```

Main program after conversion:

```
IF *BROWSER-IO NE 'RICHGUI'  
  INPUT USING MAP 'MMENU'  
ELSE  
  PROCESS PAGE USING 'AMENU'  
END-IF
```

The `INPUT` statement can come with a message text that is displayed in the status bar. There is no direct replacement for this construction because the `PROCESS PAGE` statement (in contrast to the `PROCESS PAGE UPDATE` statement) does not support the `SEND EVENT` clause.

Example 2

Main program before conversion:

```
INPUT WITH TEXT MSG01 USING MAP 'MMENU'
```

Main program after conversion (no message will be displayed):

```
IF *BROWSER-IO NE 'RICHGUI'  
  INPUT WITH TEXT MSG01 USING MAP 'MMENU'  
ELSE  
  PROCESS PAGE USING 'AMENU'  
END-IF
```

REINPUT Statement

The replacement for the `REINPUT` statement is the `PROCESS PAGE UPDATE` statement. In its simplest form, the `REINPUT` statement comes with a message text that is displayed in the status bar. In the converted code, this is handled by the `SEND EVENT` clause of the `PROCESS PAGE UPDATE` statement.

Example 1

Main program before conversion:

```
REINPUT [FULL] WITH TEXT MSG01
```

Main program after conversion:

```
IF *BROWSER-IO NE 'RICHGUI'
  REINPUT [FULL] WITH TEXT MSG01
ELSE
  PROCESS PAGE UPDATE [FULL]
  AND SEND EVENT 'nat:page.message'
  WITH PARAMETERS
    NAME 'type' VALUE 'E'
    NAME 'short' VALUE MSG01
  END-PARAMETERS
END-IF
```

The `REINPUT` statement can come with a message number and replacements. In this case, the message must be created from number and replacements before it is sent to the status bar with the `SEND EVENT` clause.

Example 2

This example uses a subprogram `GETMSTXT` that builds the message text from number and replacements.

Main program before conversion:

```
REINPUT [FULL] WITH TEXT *MSGNR, REPL1, REPL2
```

Main program after conversion:

```
IF *BROWSER-IO NE 'RICHGUI'
  REINPUT [FULL] WITH TEXT *MSGNR, REPL1, REPL2
ELSE
  CALLNAT 'GETMSTXT' MSTEXT MSGNR REPL1 REPL2
  PROCESS PAGE UPDATE [FULL]
  AND SEND EVENT 'nat:page.message'
  WITH PARAMETERS
    NAME 'type' VALUE 'E'
    NAME 'short' VALUE MSTEXT
  END-PARAMETERS
END-IF
```

Example 3

The `REINPUT` statement can come with a `MARK` clause in order to put the focus on a field. This case requires that a `statusprop` property is created for the field during map conversion. The variable bound to the `statusprop` property is then used before the `PROCESS PAGE UPDATE` statement to set the `FOCUS` to the field.

Main program before conversion:

```
REINPUT [FULL] WITH TEXT MSG01 MARK *LIB-NAME
```

Main program after conversion:

```
01 STATUS_LIB-NAME-CV (A) DYNAMIC
...
IF *BROWSER-IO NE 'RICHGUI'
  REINPUT [FULL] WITH TEXT MSG01 MARK *LIB-NAME
ELSE
  STATUS_LIB-NAME-CV := 'FOCUS'
  PROCESS PAGE UPDATE FULL
  AND SEND EVENT 'nat:page.message'
  WITH PARAMETERS
    NAME 'type' VALUE 'W'
    NAME 'short' VALUE MSG01
  END-PARAMETERS
END-IF
```

PF-Key Event Handling

The original application might contain checks for the content of the system variable *PF-KEY at arbitrary places in the code. In order to handle function key events correctly in the converted application, several things need to be achieved:

- In response to the function keys, the converted application must raise events that are named like the possible contents of *PF-KEY. This can be achieved by using a page template such as *NATPAGEHOTKEYS_TEMPLATE.xml* which contains the required hotkey definitions.
- A common local variable must be set up right after the `INPUT` or `PROCESS PAGE` statement that contains either the value *PF-KEY or *PAGE-EVENT, depending on the execution environment. The name of the variable can be freely chosen. In the example below, the name `XEVENT` is used.
- The event `nat:page.end` must be handled in such a way so that the program terminates. This event is raised when the user leaves the page or closes the browser session.
- A default event handler must be set up that takes care of the values of *PAGE-EVENT that are not expected by the original application code. These unexpected events are simply replied with a `PROCESS PAGE UPDATE FULL` statement.

Example

```

01 XEVENT (U) DYNAMIC
...
PROCESS PAGE USING ...
...
IF *BROWSER-IO = 'RICHGUI'
  DECIDE FOR FIRST CONDITION
    WHEN *PAGE-EVENT = 'nat:page.end'
      STOP
    WHEN *PAGE-EVENT = MASK ('PF'*) OR = MASK ('PA'*)
      OR = 'ENTR' OR = 'CLR'
      XEVENT := *PAGE-EVENT
    WHEN NONE
      PROCESS PAGE UPDATE FULL
  END-DECIDE
ELSE
  XEVENT := *PF-KEY
END-IF

```

All references to `*PF-KEY` in the code must then be replaced by references to `XEVENT`.

SET KEY Statement

Natural for Ajax provides two controls ([NJX:BUTTONITEMLIST](#) and [NJX:BUTTONITEMLIST-FIX](#)) that represent a row of buttons. These controls can be used to replace the visual representation of the function keys from the original application. If the page template *NATPAGEPFKEYS_TEMPLATE.xml* or a similar individually adapted template is used during map conversion, each resulting page will contain a row of function key buttons. The subject of this section is how the converted application can control the labeling and the program-sensitivity of the function keys with only little code changes.

Natural controls the labeling and program-sensitivity of the function keys in a highly dynamic way. The corresponding application code (`SET KEY` statements) can be distributed across program levels and can be lexically separated from the corresponding `INPUT` statements. Also, the `SET KEY` statement has several flavors, some affecting all keys and others affecting only individual keys. As a result, the status of the function keys at a given point in time can only be determined at application runtime.

Therefore, the following approach is chosen: Natural provides the application programming interface (API) `USR4005` that reads the current function key naming and program-sensitivity at runtime. During code conversion, a call to this API is inserted after each `SET KEY` statement or into each round trip. This call reads the function key status and passes it to the user interface.

Example

Main program before conversion:

```
SET KEY ENTR NAMED 'Enter' PF1 NAMED 'F1' PF2 NAMED 'F2'
PF3 NAMED 'Modify' PF4 NAMED 'Delete' PF5 NAMED 'F5'
PF6 NAMED 'F6' PF7 NAMED 'Create' PF8 NAMED 'Display'
PF9 NAMED 'F9' PF10 NAMED 'F10' PF11 NAMED 'F11' PF12 NAMED 'F12'
*
INPUT USING MAP "KEYS-M"
*
END
```

Map before conversion:

```

*** PF-Keys ***

Enter-PF1---PF2---PF3---PF4---PF5---PF6---PF7---PF8---PF9---PF10--PF11--PF12---
Enter F1    F2    Modif Delet F5    F6    Creat Displ F9    F10    F11    F12

```

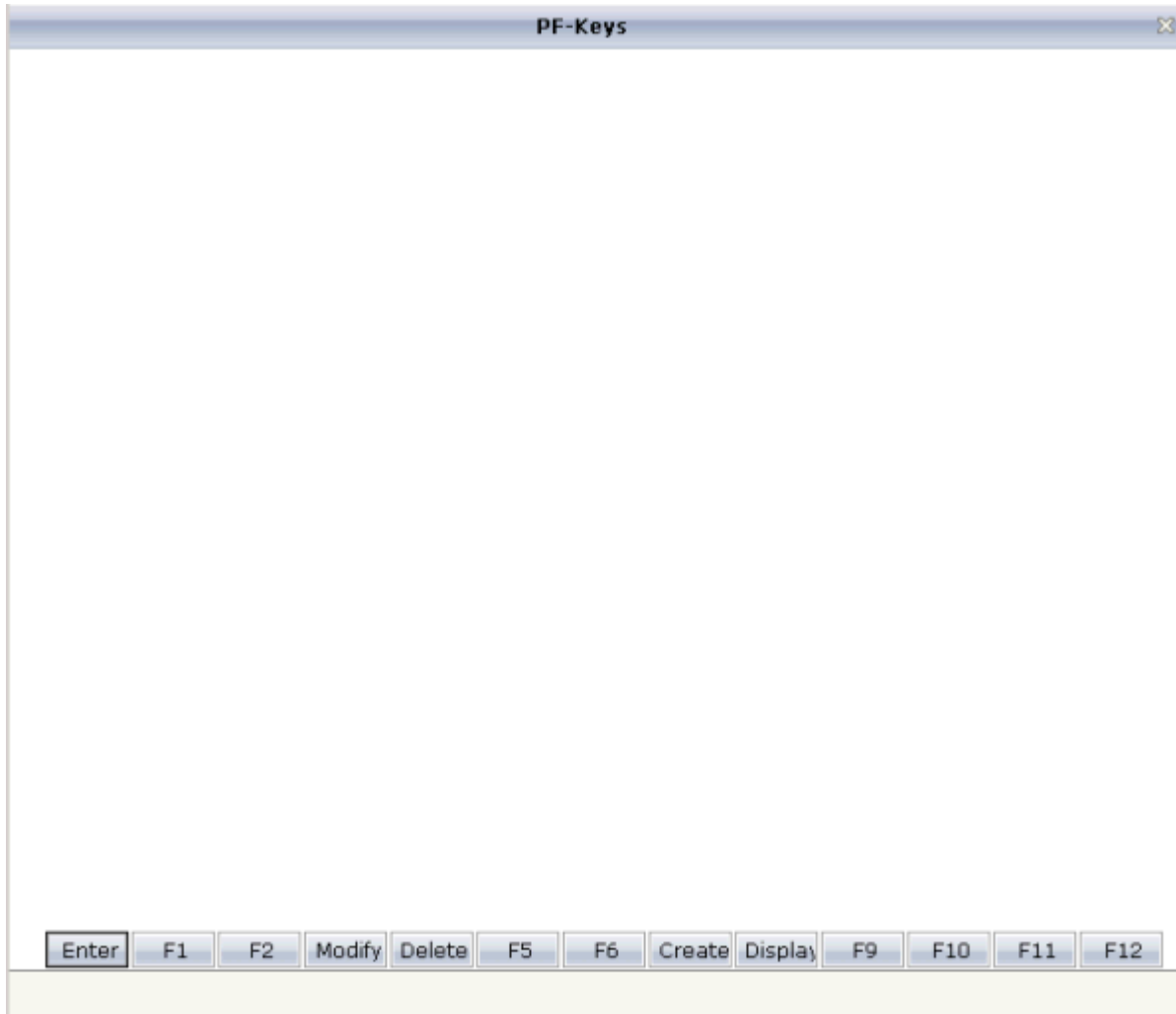

Main program after conversion:

```

DEFINE DATA LOCAL
1 PFKEY (1:*)
2 METHOD (A) DYNAMIC
2 NAME (A) DYNAMIC
2 TITLE (A) DYNAMIC
2 VISIBLE (L)
1 METHODS (A4/13) CONST <'ENTR','PF1','PF2','PF3','PF4',
'PF5','PF6','PF7','PF8','PF9','PF10','PF11','PF12'>
END-DEFINE
*
SET KEY ENTR NAMED 'Enter' PF1 NAMED 'F1' PF2 NAMED 'F2'
PF3 NAMED 'Modify' PF4 NAMED 'Delete' PF5 NAMED 'F5'
PF6 NAMED 'F6' PF7 NAMED 'Create' PF8 NAMED 'Display'
PF9 NAMED 'F9' PF10 NAMED 'F10' PF11 NAMED 'F11' PF12 NAMED 'F12'
*
IF *BROWSER-IO NE "RICHGUI"
    INPUT USING MAP "KEYS-M"
ELSE
    EXPAND ARRAY PFKEY TO (1:13)
    METHOD(1:13) := METHODS (*)
    CALLNAT "GETKEY-N" PFKEY (*)
    PROCESS PAGE USING "KEYS-A"
END-IF
*
END

```

Page after conversion:



Explanation

The structure `PFKEY` is generated into the Natural adapter of the page as the application interface to the `BUTTONITEMLISTFIX` control.

The subprogram `GETKEY-N` is a convenience wrapper for the API subprogram `USR4005`. It uses `USR4005` to determine the labeling and the program-sensitivity status for a given list of function keys. Each function key is identified by the `*PF-KEY` value it raises. `GETKEY-N` returns the function key information in a data structure suitable for the application interface of the `BUTTONITEMLISTFIX` control. The subprogram is delivered in the library `SYSEXNJX` in source code and can be adapted to the needs of the application.

Processing Rules

The Natural maps in the application to be converted may contain processing rules. In the sense of a Natural for Ajax application, the processing rules are server-side validations because they are executed on the Natural server side of the application.

In order to extract processing rules from the maps and to turn them into server-side validations in the converted application, the Natural Engineer function “Separate Processing Rules from Maps” can be used.

There is currently no function available that automatically turns processing rules into client-side validations in Application Designer.

System Variables

If a map displays a system variable (for example, *DATX), a specific default conversion rule takes care that the necessary code for handling the system variable is generated into the Natural adapter of the resulting page layout.

Example 1

The map displays the contents of the system variables *DATX and *TIMX. The contents of these system variables are not modifiable.

The `DEFINE DATA` statement of the adapter will then contain:

```
LOCAL
01 XDATX (A8)
01 XTIMX (A8)
```

The body of the adapter will then contain:

```
XDATX := *DATX
XTIMX := *TIMX
*
PROCESS PAGE ... WITH
PARAMETERS
...
NAME U'XDATX'
VALUE XDATX
NAME U'XTIMX'
VALUE XTIMX
END-PARAMETERS
```

The main program needs no special adaptation.

Example 2

The map displays the content of the system variable *CODEPAGE. The content of this system variables is modifiable.

The `DEFINE DATA` statement of the adapter will then contain:

```
LOCAL  
01 XCODEPAGE (A64)
```

The body of the adapter will then contain:

```
XCODEPAGE := *CODEPAGE  
*  
PROCESS PAGE ... WITH  
PARAMETERS  
...  
  NAME U'XCODEPAGE'  
  VALUE XCODEPAGE  
...  
END-PARAMETERS  
*  
*CODEPAGE := XCODEPAGE
```

The main program needs no special adaptation.

Variable Names Containing Special Characters

A similar procedure applies to special characters contained in variable names. These are the following special characters:

+

/
@
\$
&
\$



Note: The hash (#) can occur only as the first character.

Variables names containing these special characters cannot be directly bound to Application Designer control attributes. A specific default conversion rule replaces the names containing these

special characters with configurable replacements. The original field name is generated into the parameter data area of the Natural adapter and a corresponding mapping is generated into the `PROCESS PAGE` statement of the adapter.

Example

The map displays the variables `#FIRST` and `#LAST`.

The `DEFINE DATA` statement of the adapter will then contain:

```
DEFINE DATA PARAMETER
1 #FIRST (A16)
1 #LAST (A20)
```

The body of the adapter will then contain:

```
...
PROCESS PAGE ... WITH
PARAMETERS
...
NAME U'HFIRST'
  VALUE #FIRST
NAME U'HLAST'
  VALUE #LAST
...
END-PARAMETERS
```

The main program needs no special adaptation.

28

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**

29

Positioning of Controls inside a Container

■ Row Types - TR and ITR	188
■ Some More Details on ITR	189
■ TR Properties	191
■ ITR Properties	192

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:

The image shows two examples of form layouts. The top example, titled 'With TR', shows two rows. In the first row, the 'First Name' label is followed by a text input field. In the second row, the 'Last Name' label is followed by a text input field. The input fields are aligned to the right, and the labels are aligned to the left. The bottom example, titled 'With ITR', shows two rows. In the first row, the 'First Name' label is followed by a text input field. In the second row, the 'Last Name' label is followed by a text input field. The input fields are aligned to the right, and the labels are aligned to the left. The layout is more flexible than the TR example.

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.

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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(A) You do not define a height at all. As consequence the height of the control will follow the height of its content.</p> <p>(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".</p> <p>(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.</p>	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p>	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.		
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. 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.	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 specifies the position of the control inside the column.		middle bottom
fixlayout	<p>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.</p> <p>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.</p> <p>You typically use this control if the content of the control's area is flexibly sizeable. E.g. if the content (e.g. a TEXTGRID control) is following the size of the container.</p> <p>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.</p>	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
itrclass	CSS style class definition that is directly passed into this control.	Optional	

	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.		
tablestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
Binding			
itrstyleprop	\$en/popupwizard/njx__attr_itrstyleprop\$	Optional	

30

Defining the Width of Controls inside a Container

■ Controlling the Width of Controls	196
■ HDIST and VDIST Controls	198
■ HDIST Properties	201
■ VDIST Properties	201
■ rowspan and colspan Definitions	202
■ CELLSPAN Control	203
■ CELLSPAN Properties	204
■ Rules for Positioning Controls inside Containers	206

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, put either an absolute pixel (`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 absolute width values:

```
<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>
```

For two different screen sizes, the HTML page looks as follows:

First Name	<input type="text"/>
Last Name	<input type="text"/>

Factor1	<input type="text"/>
Factor2	<input type="text"/>

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:

Factor1	<input type="text" value="0"/>
Factor2	<input type="text" value="0"/>

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:



The screenshot shows a web form with a title bar labeled "HDIST Example". Below the title bar, there is a label "Zip Code / Town" followed by two adjacent text input fields. The first input field is for the zip code and the second is for the town name. The two input fields are separated by a small, consistent horizontal gap, which is the HDIST control.

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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	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>
```

```
</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:

Cellspan Example

Factor 1

0

Factor 1

0

Factor 2

0

☐ Activated

☐ Generated

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	<div>Width of the control.</div> <div>There are three possibilities to define the width:</div> <div>(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.</div> <div>(B) Pixel sizing: just input a number value (e.g. "100").</div> <div>(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.</div>	Optional	<div>100</div> <div>120</div> <div>140</div> <div>160</div> <div>180</div> <div>200</div> <div>50%</div> <div>100%</div>
height	<div>Height of the control.</div> <div>There are three possibilities to define the height:</div> <div>(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.</div>	Optional	<div>100</div> <div>150</div> <div>200</div> <div>250</div> <div>300</div>

	<p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>		<p>250</p> <p>400</p> <p>50%</p> <p>100%</p>
title	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
colspanprop	\$en/popupwizard/njx__attr_colspanprop\$	Optional	
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
cellstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p>

	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
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.

31

Vertical Sizing of Containers and Controls

■ Vertical Pixel Sizing	208
■ Vertical Percentage Sizing	209
■ Finishing the Example	212

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.

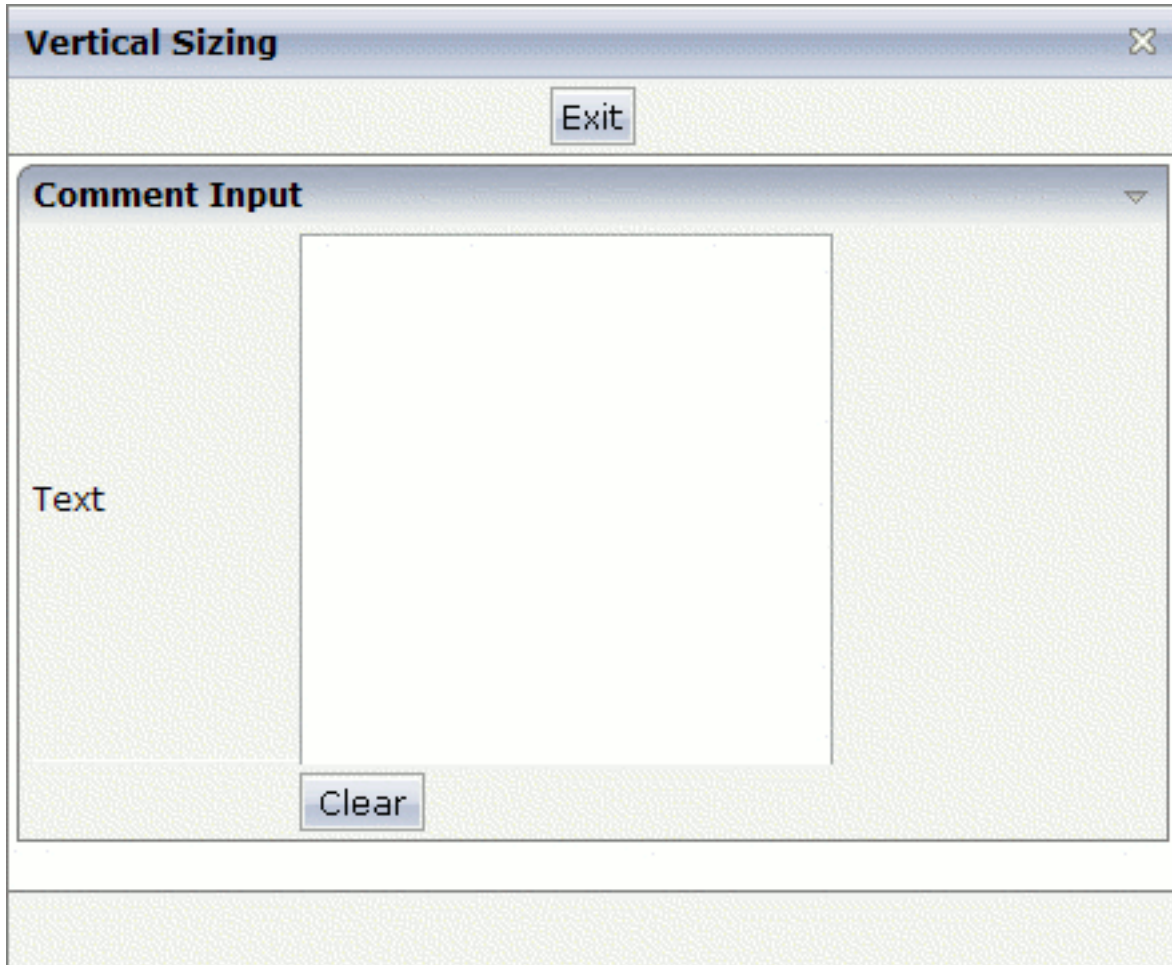
Vertical Pixel Sizing

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.

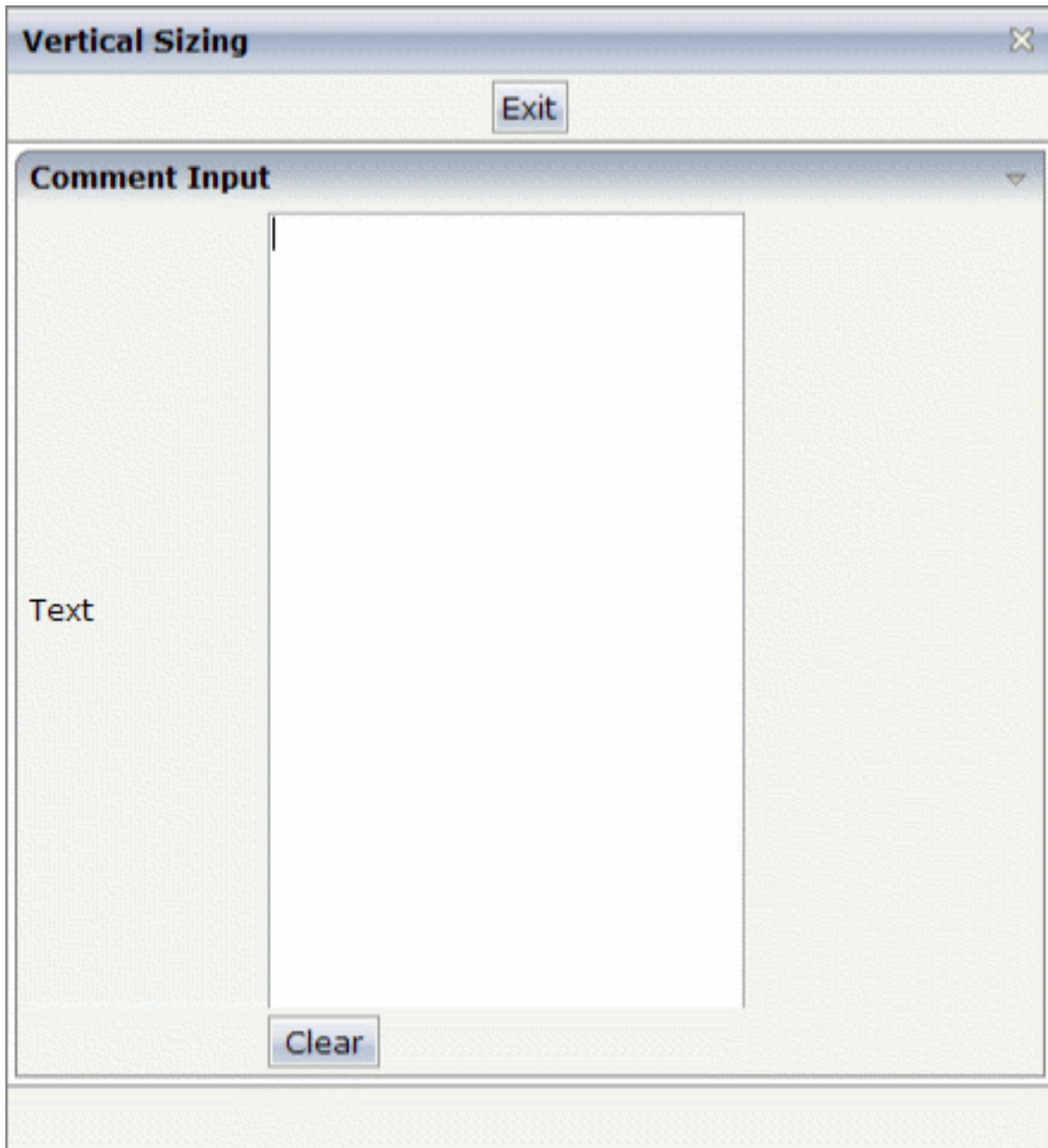
The XML layout definition looks as follows:

```
<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%"`.

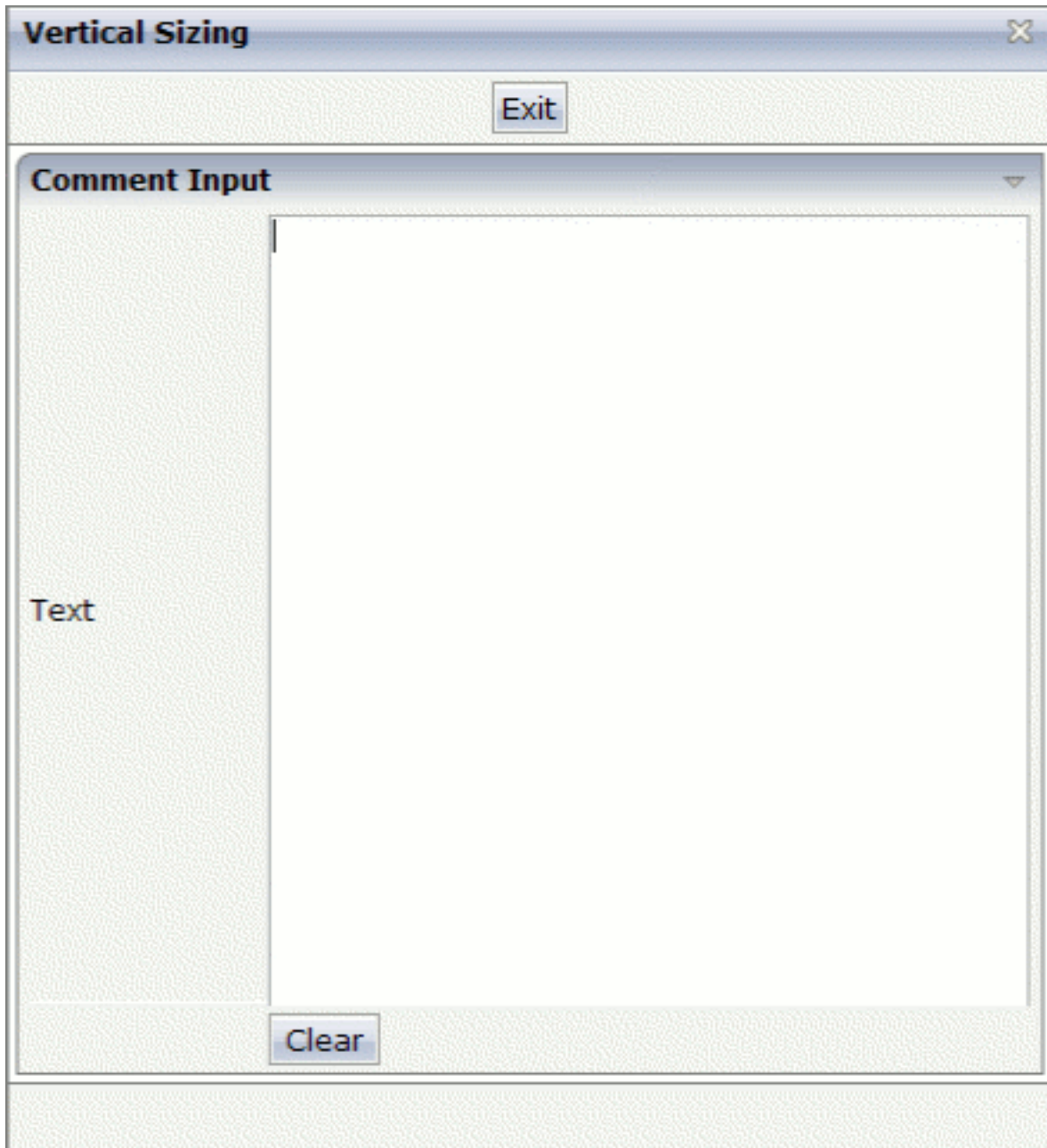
Finishing the Example

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:



32

Overview of Different Containers

■ Different Kind of Containers	216
■ Row Containers	216
■ Column Containers	217
■ Row and Column Containers in Combination	218
■ Nesting Containers	219

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:

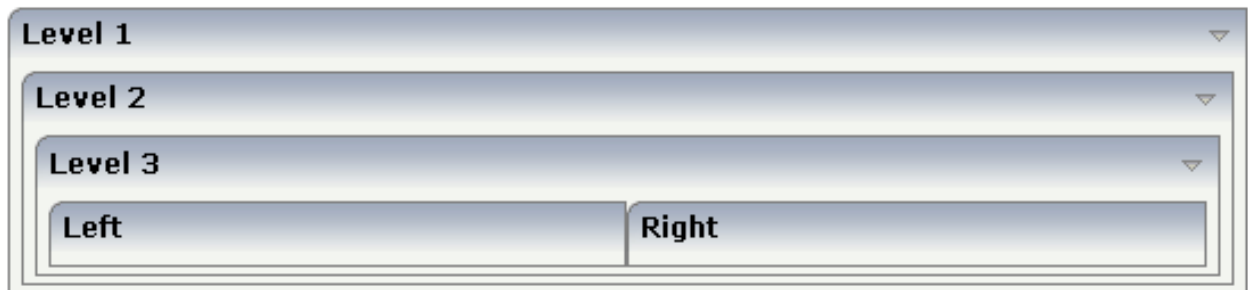


Nesting Containers

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:



33 ROWAREA and COLAREA

■ ROWAREA Properties	222
■ COLAREA Properties	229

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	\$en/popupwizard/njx__attr_nameprop\$	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	\$en/popupwizard/njx_area_attr_foldedprop\$	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 want to pass one changed value to all its representaion 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			

image	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	
imageprop	\$en/popupwizard/njx__attr_titleimageprop\$	Optional	
withtoppadding	<p>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.</p> <p>By specifying "false" you can avoid this behaviour. "</p>	Optional	true false
withleftborder	<p>The control normally renders a black border around its area. With the properties WITHLEFTBORDER, WITHRIGHTBORDER and WITHBOTTOMBORDER you can avoid this.</p> <p>Reason behing: somtimes 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.</p>	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	<p>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 is part of the normal tab-sequence of a page.</p> <p>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.</p>	Optional	true false

fixlayout	<p>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.</p> <p>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.</p> <p>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.</p> <p>When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible.</p> <p>- 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.</p>	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	<p>The control by default renders some blank vertical space (3 pixels) on bottom of the content area.</p> <p>By specifying "false" you can avoid this behaviour.</p>	Optional	true false
withcontentbottompadding	The control by default renders some blank vertical space (3 pixels) on bottom of the content area.	Optional	true false

	By specifying "false" you can avoid this behaviour.		
withfadedtoggling	<p>The animation of the controls can be switched off! Please take a look in your ciscoconfig.xml file. Set animatecontrols="true" (default) if you generally want to animate all of your controls.</p> <p>The rowarea control has a seperate switch (withfadedtoggling = true/false) to (de)activate the 'FadedToggling' animation especially for this single rowarea control.</p> <p>Notice: Entering true or false into the withfadedtoggling attribute overwrites the general animatecontrols setting !</p>	Optional	true false
stylevariant	<p>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.</p> <p>Purpose: you can set up style variants in the style sheet defintion and use them multiple times by addressing them via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	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	<p>Multi language dependent text that is displayed as tooltip of the toggle image.</p> <p>Do not specify a "toogleimagetitle" inside the control if specifying a "toggleimagetextid".</p>	Optional	
Online Help			
title	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

Natural			
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: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: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.	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: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 for a generated statusprop variable to which field the statusprop belongs.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control. 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 control.	Optional	

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	\$en/popupwizard/njx__attr_nameprop\$	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.	Optional	true false

	<p>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'.</p> <p>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.</p>		
image	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	
imageprop	\$en/popupwizard/njx__attr_titleimageprop\$	Optional	
fixlayout	<p>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.</p> <p>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.</p> <p>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.</p>	Optional	true false

	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 being: 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.	Optional	background-color: #FF0000

	<p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		<p>color: #0000FF</p> <p>font-weight: bold</p>
contenttablestyle	CSS style that is applied to the content are of the COLAREA control.	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
withcontenttoppadding	<p>The control by default renders some blank vertical space (3 pixels) on bottom of the content area.</p> <p>By specifying "false" you can avoid this behaviour.</p>	Optional	<p>true</p> <p>false</p>
withcontentbottompadding	<p>The control by default renders some blank vertical space (3 pixels) on bottom of the content area.</p> <p>By specifying "false" you can avoid this behaviour.</p>	Optional	<p>true</p> <p>false</p>
titlerowontop	Default value is 'true'. If set to 'false' the titlerow is rendered at the bottom of this area.	Optional	<p>true</p> <p>false</p>
stylevariant	<p>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.</p> <p>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 offerst two variants</p>	Optional	

	"VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!		
withtoppadding	<p>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.</p> <p>By specifying "false" you can avoid this behaviour. "</p>	Optional	<p>true</p> <p>false</p>
Online Help			
title	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

34 ROWAREAWITHHEADER

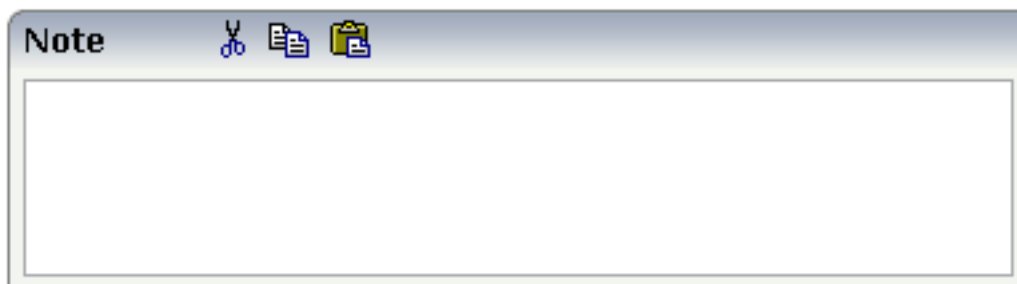
■ Simple Example	236
■ ROWAREAWITHHEADER Properties	237
■ ROWAREAHEADER Properties	239
■ ROWAREABODY Properties	240

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).

ROWAREAWITHHEADER Properties

Basic			
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>Name of the adapter parameter that dynamically controls whether clicking on the title of the area will fold/unfold this area.</p> <p>Valid values provided by the adapter parameter are TRUE (=foldable) and FALSE(=not foldable).</p>	Optional	
foldedprop	\$en/popupwizard/njx_area_attr_foldedprop\$	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	
Appearance			
height	(already explained above)		
withleftborder	<p>The control normally renders a black border around its area. With the properties WITHLEFTBORDER, WITHRIGHTBORDER and WITHBOTTOMBORDER you can avoid this.</p> <p>Reason being: sometimes you want a ROWAREA/COLAREA to be used as "neighbour" of other ROWAERA/COLAREA controls. In this</p>	Optional	true false

	case one of the "neighbours" has 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
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	\$en/popupwizard/njx__attr_titleimageprop\$	Optional	
nameprop	\$en/popupwizard/njx__attr_nameprop\$	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.	Optional	true false

	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.		
--	---	--	--

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 lanaguage 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 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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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.	Optional	true false

	By specifying "false" you can avoid this behaviour.		
withcontentbottompadding	<p>The control by default renders some blank vertical space (3 pixels) on bottom of the content area.</p> <p>By specifying "false" you can avoid this behaviour.</p>	Optional	true false

35

ROWTABAREA and COLTABAREA

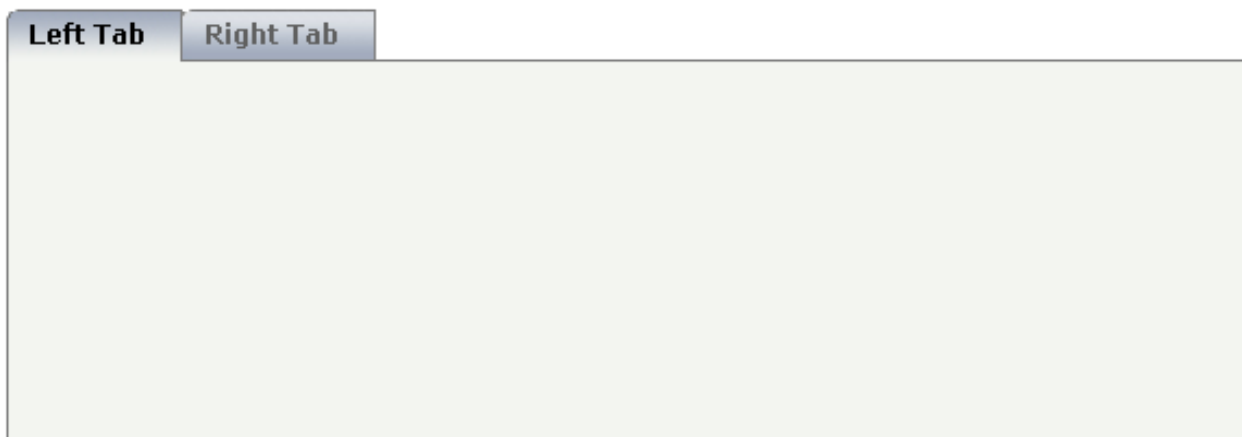
▪ ROWTABAREA Properties	245
▪ COLTABAREA Properties	258
▪ TABPAGE Properties	588
▪ The Most Common Error	267
▪ Example: Controlling which Tab is displayed by the Server Adapter	267
▪ Example: Controlling the Visibility of Tab Pages	268

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 characters. 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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>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.</p> <p>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.</p>	Obligatory	

withclose1	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
withclose2	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
withclose3	<p>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.</p> <p>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.</p>	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	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	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	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	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	Optional	

	<p>"healthy" id: starting with characters, without blanks and without "strange" characters.</p> <p>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.</p>		
withclose6	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
withclose7	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	

withclose8	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
withclose9	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
withclose10	<p>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.</p> <p>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.</p>	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	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	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	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	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	Optional	

	<p>"healthy" id: starting with characters, without blanks and without "strange" characters.</p> <p>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.</p>		
withclose13	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
withclose14	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	

withclose15	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	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	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	true false
Binding			
openedindexprop	\$en/popupwizard/njx_rowtabarea_attr_openedindexprop\$	Optional	
openmethod	\$en/popupwizard/njx_rowtabarea_attr_openmethod\$	Optional	
visibleprop1	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop2	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop3	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop4	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop5	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop6	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop7	Name of property that defines if the corresponding tag is visible or not.	Optional	

visibleprop8	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop9	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop10	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop11	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop12	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop13	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop14	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop15	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop16	Name of property that defines if the corresponding tag is visible or not.	Optional	
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 offerst 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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	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. 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.	Obligatory	

	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.		
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"	Optional	

	<p>id: starting with characters, without blanks and without "strange" characters.</p> <p>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.</p>		
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	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	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	<p>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.</p> <p>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.</p>	Optional	
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. 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	
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. 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	
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	\$en/popupwizard/njx_rowtabarea_attr_openedindexprop\$	Optional	
openmethod	\$en/popupwizard/njx_rowtabarea_attr_openmethod\$	Optional	
visibleprop1	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop2	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop3	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop4	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop5	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop6	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop7	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop8	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop9	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop10	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop11	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop12	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop13	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop14	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop15	Name of property that defines if the corresponding tag is visible or not.	Optional	
visibleprop16	Name of property that defines if the corresponding tag is visible or not.	Optional	
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	Optional	VAR1

	<p>style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>		
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 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	

TABPAGE Properties

Basic			
id	Id of the TABPAGE. Each page has a certain id that reference to the PAGE1 .. PAGE9 definition you do inside the ROW/COLTABAREA control that contains the TABPAGE. If clicking onto a "tab" then the TABPAGE with the associated id is displayed as consequence.	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 ot "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. 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.	Optional	true false

	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 ROWTABAREA level:

The screenshot shows a user interface for a ROWTABAREA control. At the top, there is a title bar that says "Dynamic setting of index in TABAREA". Below this, there is a label "Index" followed by a dropdown menu currently showing "Second (=1)". Underneath the dropdown are three tab buttons labeled "First", "Second", and "Third". The "Second" tab is highlighted, indicating it is the active tab. The main content area below the tabs is a large, empty rectangular box.

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 screenshot shows a web application with a tabbed interface. The tabs are labeled "Rich", "User", "Interfaces", "for", "Business", and "Applications". The "Rich" tab is currently selected, and its content area displays the word "Rich". Below the tabs is a "Visibility" section, which includes a dropdown arrow and a row of checkboxes for each tab: ☒ Rich, ☒ User, ☒ Interfaces, ☒ for, ☒ Business, and ☒ Applications.

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>
  ...
  ...
  ...
</rowtabarea>

<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.

36 ROWTABLE0 and COLTABLE0

■ ROWTABLE0 Properties	273
■ COLTABLE0 Properties	274

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:



The screenshot shows a web form with a title bar labeled "Area 1". Inside the form, there are two input fields. The first field is labeled "Factor 1" and contains the value "0". The second field is labeled "Factor 2" and also contains the value "0". The form has a light green background and a thin border.

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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
align	<p>Alignment of the content of the ITR row.</p> <p>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.</p> <p>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.</p>	Optional	left center right
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	top middle bottom
fixlayout	<p>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.</p> <p>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.</p>	Optional	true false

	<p>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.</p> <p>When using vertical percentage based sizing you should pay attention to set the fixlayout-property to "true" as often as possible.</p> <ul style="list-style-type: none"> - 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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
flashprop	\$en/popupwizard/njx__attr_flashprop\$	Optional	

COLTABLE0 Properties

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

Basic			
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p>	Optional	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p>

	(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%
widthprop	\$en/popupwizard/njx_coltable0_attr_widthprop\$	Optional	
takefullheight	<p>Indicates if the content of the control's area gets the full available height.</p> <p>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'.</p> <p>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.</p>	Optional	true false
fixlayout	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>	Optional	true false
tablestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p>	Optional	background-color: #FF0000 color: #0000FF

	<p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		font-weight: bold
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

37

ROWDYNAVIS and COLDYNAVIS

■ ROWDYNAVIS Properties	279
■ COLDYNAVIS Properties	280
■ Some Comments on Controlling the Visibility of Controls	282

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.



The screenshot shows a web component titled "Address Input" with a dropdown arrow in the top right corner. Inside the container, there is a label "Country" followed by a text input field containing the text "Germany".

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



The screenshot shows the same "Address Input" web component. Now, there are two input fields. The first is labeled "Country" and contains "United States". Below it is a second field labeled "State" which contains "California".

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	\$en/popupwizard/njx_dynavis_attr_valueprop\$	Obligatory	
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
style	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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 requires to have specified the height and the width (if available as property) of the control.	Optional	true false

	<p>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.</p> <p>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.</p> <p>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.</p>		
--	---	--	--

COLDYNAVIS Properties

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

Basic			
valueprop	\$en/popupwizard/njx_dynavis_attr_valueprop\$	Obligatory	
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	100 120 140 160 180 200 50% 100%
takefullheight	<p>Indicates if the content of the control's area gets the full available height.</p> <p>If you use percentage sizing inside the control's area then this property must be switched to 'true'. If you use no explicit</p>	Optional	true false

	<p>vertical sizing at all - or you use vertical pixel sizing for your controls - the property must be switched to 'false'.</p> <p>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.</p>		
style	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
fixlayout	<p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>	Optional	<p>true</p> <p>false</p>
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	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 visibility 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.

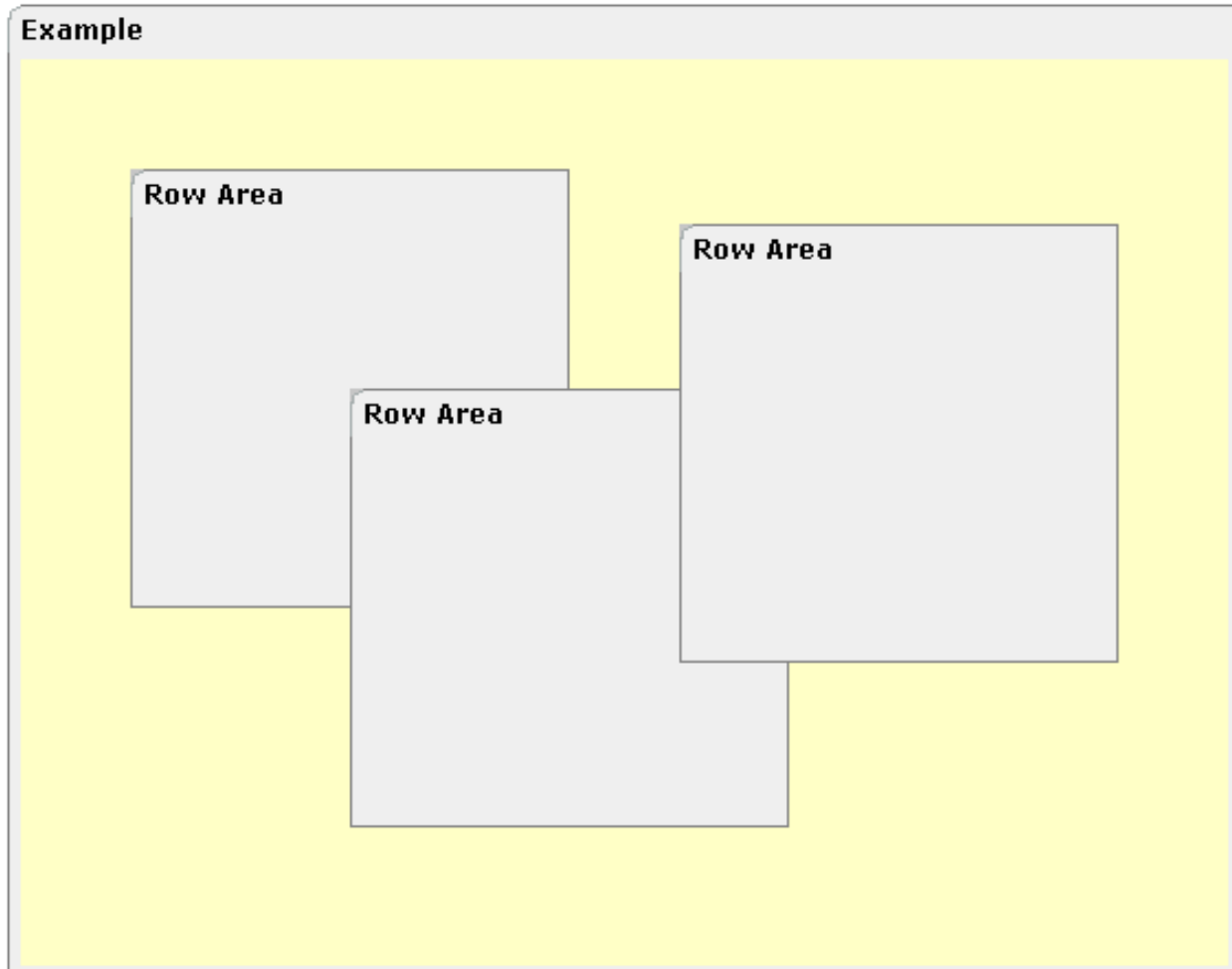
38

ROWDIV and INNERDIV

■ When to Use ROWDIV and INNERDIV Containers	286
■ ROWDIV Properties	287
■ INNERDIV Properties	288

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. INNERDIV 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:

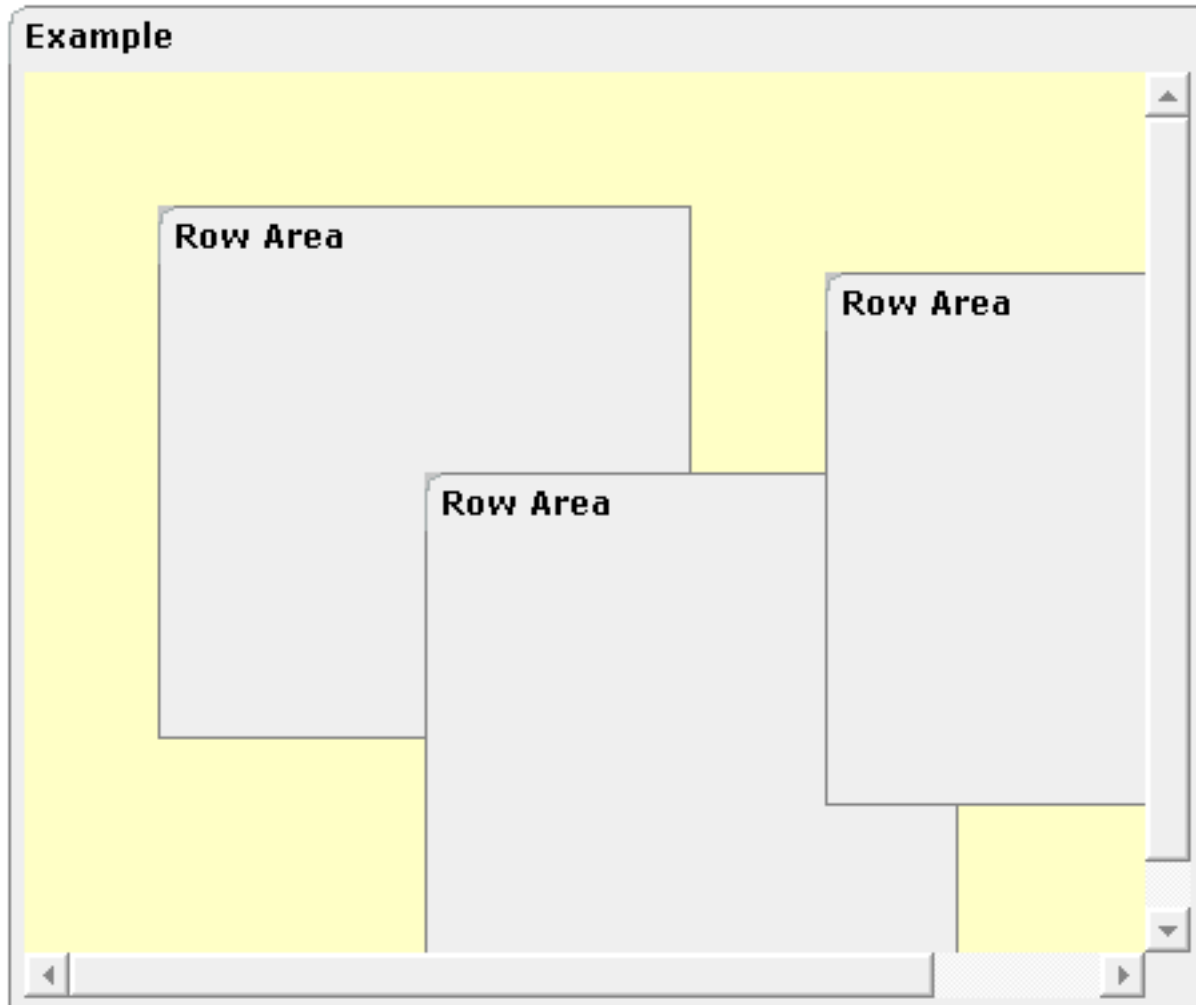


Inside a ROWAREA container, a ROWDIV container is arranged. Inside the ROWDIV container, three INNERDIV containers are arranged - each one holding a ROWAREA.

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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Obligatory	100
			150
			200
			250
			300
			250
			400
			50%
			100%
style	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	
divclass	<p>CSS style class definition that is directly passed into this control.</p> <p>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.</p>	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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Obligatory	100
			120
			140
			160
			180
			200
			50%
			100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Obligatory	100
			150
			200
			250
			300
			250
			400
			50%
			100%
left	Left position of control. Either define a pixel value ("100") or a percentatge value ("30%").	Obligatory	
top	Top position of control. Either define a pixel value ("100") or a percentatge 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

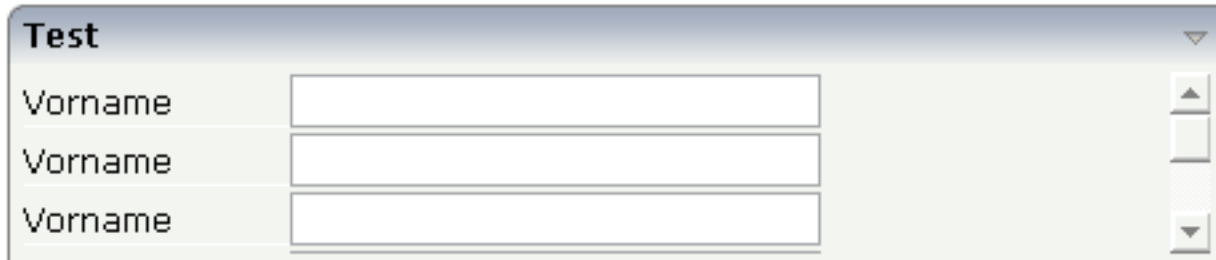
			int-value
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
lefthdistance	If set to "true" then a small distance (3px) is kept between the left border of the control and its content. Default is "false".	Optional	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	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).	Optional	

39 ROWSCROLLAREA

■ ROWSCROLLAREA Properties	293
----------------------------------	-----

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:

A screenshot of a web browser window showing a form titled "Test". Inside the form, there is a container area (ROWSCROLLAREA) that contains three identical rows. Each row consists of a label "Vorname" followed by a text input field. The container area is smaller than the total height of the three rows, which causes a vertical scrollbar to appear on the right side of the container. The scrollbar has a small handle in the middle, indicating that the content is scrollable.

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>
  </rowscrollarea>
</rowarea>
```

```

        </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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Obligatory	100 150 200 250 300 250 400 50% 100%
takefullheight	<p>Indicates if the content of the control's area gets the full available height.</p> <p>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'.</p> <p>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</p>	Optional	true false

	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
areaclass	<p>CSS style class definition that is directly passed into this control.</p> <p>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.</p>	Optional	
fixlayout	<p>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.</p> <p>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.</p> <p>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.</p> <p>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</p>	Optional	true false

	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	
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

40

HSPLIT and VSPLIT

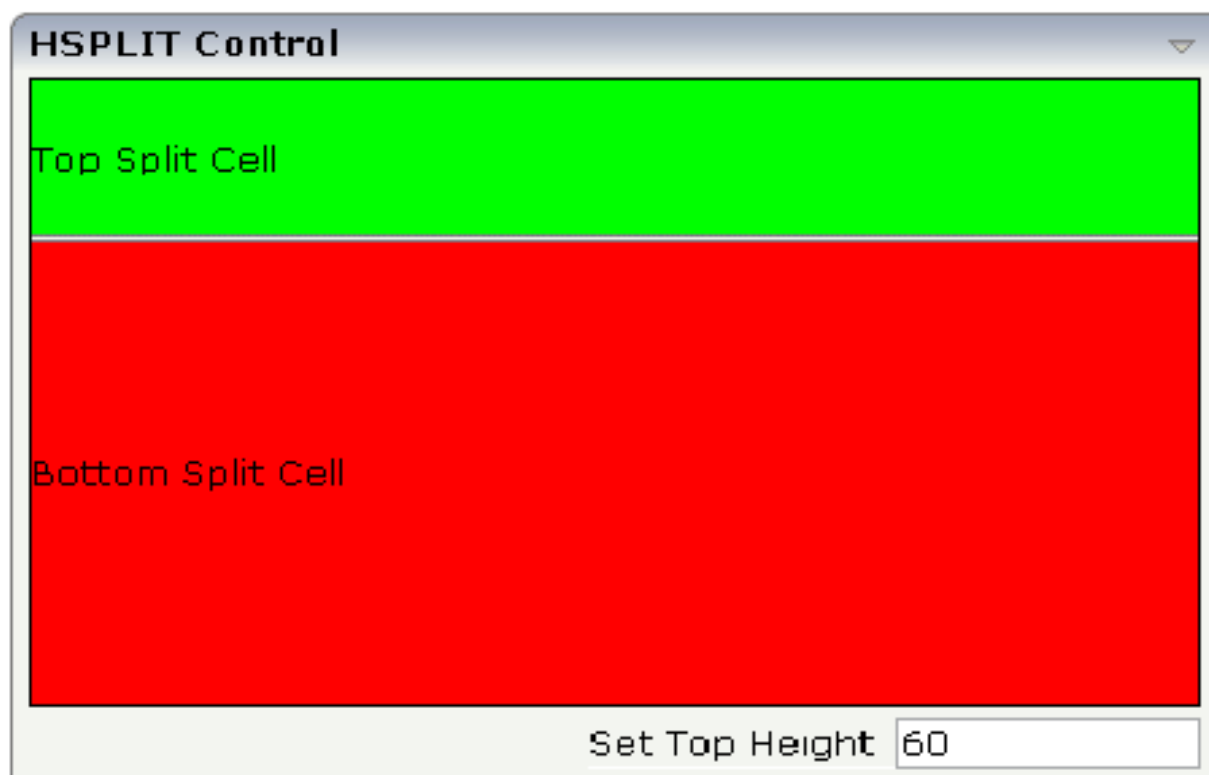
■ Example for HSPLIT	298
■ Example for VSPLIT	300
■ HSPLIT Properties	301
■ VSPLIT Properties	302
■ SPLITCELL Properties	303
■ Defining the Split Size	304

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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
heighttop	<p>Definition of the initial height of the top split area.</p> <p>The height either is a pixel value ("100") or a percentage value ("50").</p> <p>You can also define the height dynamically by your adapter - see documentation for HEIGHTTOPPROP property.</p>	Optional	1 2 3 int-value
heighttopprop	\$en/popupwizard/njx_hsplit_attr_heighttopprop\$	Optional	
hsplitstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	\$en/popupwizard/njx_vsplit_attr_widthleftprop\$	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.	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.		
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. 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

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	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	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	

Defining the Split Size

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

- Fixed definition of 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.

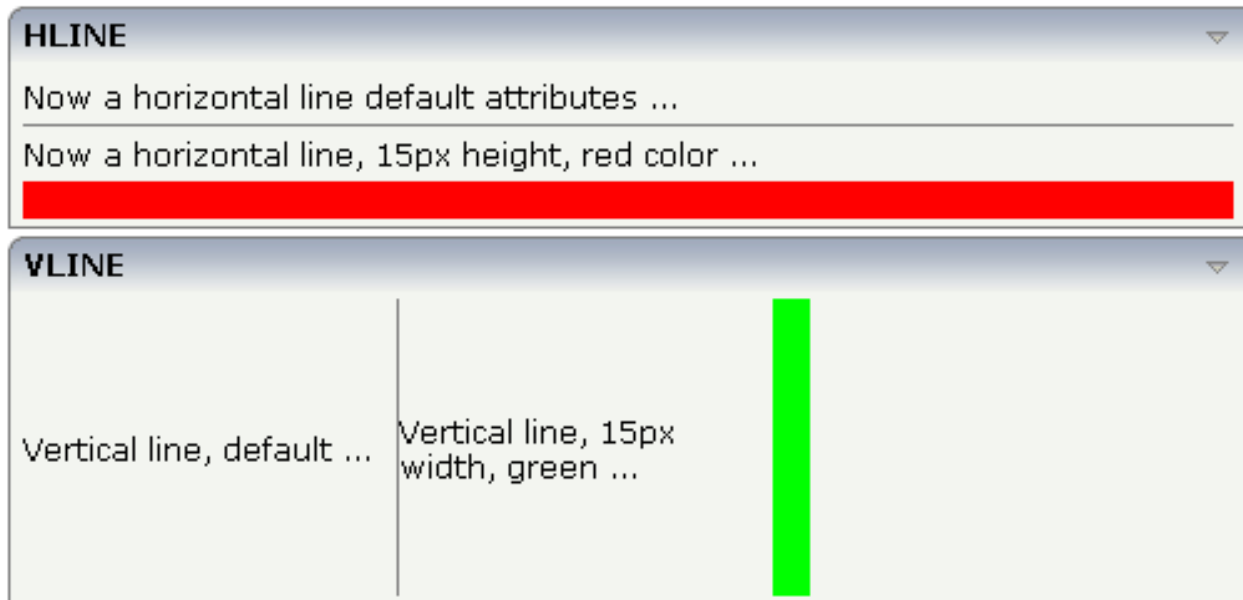
41

HLINE and VLINE

■ VLINE Properties	307
■ HLINE Properties	308

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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

42 Performance Optimization with Containers

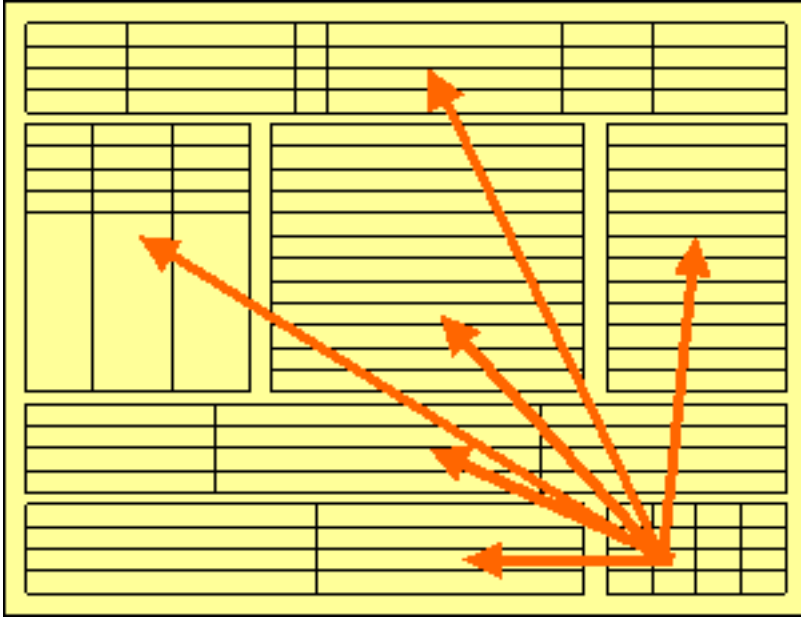
Containers internally use HTML table rendering for arranging their content: inside a container there are rows, inside the rows there are columns and inside the columns there are controls.

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:

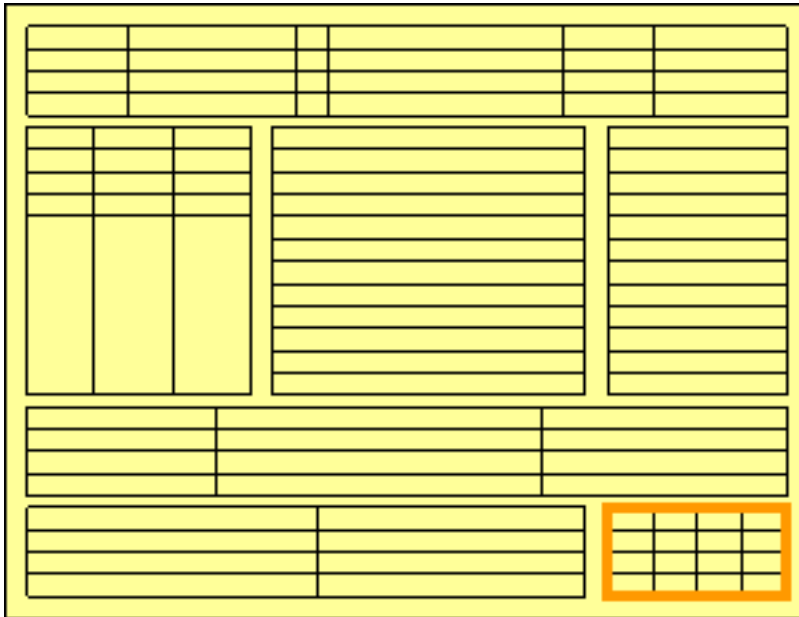
- Look into the Application Designer log file. Each server side request is recorded with its consumption of milliseconds on server side.
- Resize the page in the browser: if this is not fast but takes time, then this is an indicator for bad rendering performance - or in other words: for a lot of optimization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

43

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**
- **DROPICON**
- **FIELD**
- **FILEUPLOAD/FILEUPLOAD2**
- **ICON**
- **ICONLIST**
- **IHTML**

- IMAGEOUT
- LABEL
- MENUBUTTON
- METHODLINK
- MULTISELECT
- NEWSFEED
- RADIOBUTTON
- SCHEDULELINE
- SLIDER
- STRIPSEL
- SUBPAGE
- TABSEL
- TABSTRIP2
- TAGCLOUD
- TEXT
- TEXTOUT
- TOGGLE

Special Controls:

- ACTIVEX
- GOOGLEMAP2
- NETMEETING
- SKYPECALL

Natural for Ajax Controls:

- NJX:BUTTONITEMLIST
- NJX:BUTTONITEM
- NJX:BUTTONITEMLISTFIX

- **NJX:BUTTONITEMFIX**
- **NJX:FIELDLIST**
- **NJX:FIELDITEM**
- **NJX:FIELDVALUE**
- **NJX:NJXVARIABLE**
- **NJX:EVENTDATA**

Natural Page:

- **NATPAGE**

44

Some Common Rules for all Controls

■ Name and Text ID	318
■ Table, Row, Column, Control	318
■ Explicit Alignment	318
■ Binding to Adapter Parameters	319
■ Directly Influencing the Control Style	319
■ Dynamically Controlling the Visibility and the Display Status of Controls	320
■ Focus Management	321
■ Flushing of Inputs	322
■ Tab Sequence	322
■ Tooltips	324

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

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:

<code>displayprop</code>	<code>statusprop</code>	Control Status
FALSE (default)		EDIT
FALSE (default)	INVISIBLE	INVISIBLE
FALSE (default)	ERROR	ERROR
FALSE (default)	ERROR_NO_FOCUS	ERROR_NO_FOCUS
FALSE (default)	FOCUS	FOCUS
TRUE		DISPLAY
TRUE	INVISIBLE	INVISIBLE
TRUE	ERROR	ERROR_DISPLAY
TRUE	ERROR_NO_FOCUS	ERROR_DISPLAY
TRUE	FOCUS	DISPLAY

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 corresponding 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.

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:

Simple Tab Sequence

First Second

Third

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, a small yellow box appears showing some more detailed explanation.

The corresponding controls offer two 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..

45

BREADCRUMB

▪ Example	326
▪ Adapter Interface	326
▪ Built-in Events	326
▪ Properties	327

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

```
DEFINE DATA PARAMETER
1 ITEMS (1:*)
2 STYLE (U) DYNAMIC
2 TEXT (U) DYNAMIC
2 TOOLTIP (U) DYNAMIC
1 ITEMSINFO
2 SELECTEDITEM (I4)
END-DEFINE
```

Built-in Events

value-of-breadcrumbprop.onSelect

Properties

Basic			
breadcrumbprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
breadcrumbstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
pixeldistance	Pixel distance between the links that are rendered.	Optional	<p>1</p> <p>2</p> <p>3</p> <p>int-value</p>
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

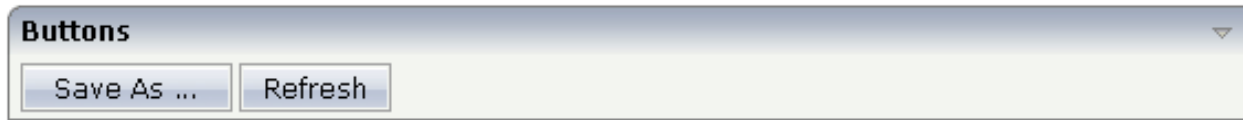
46

BUTTON

▪ Example: Simple Button	330
▪ Example: Button with Image	331
▪ Hiding and Disabling Buttons	331
▪ Properties	331

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.	Obligatory	
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	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	<p>gif</p> <p>jpg</p> <p>jpeg</p>
invisiblemode	<p>This property has three possible values:</p> <p>(1) "invisible": the button is not visible without occupying any space.</p> <p>(2) "disabled": the button is deactivated: it is "grayed" and does not show any roll over effects any more.</p> <p>(3) "cleared": the button is not visible but it still occupies space.</p>	Optional	<p>invisible</p> <p>disabled</p> <p>cleared</p>
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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</p>	Optional	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p> <p>50%</p> <p>100%</p>

	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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>CSS style definition that is directly passed into the text of this control.</p> <p>With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are:</p> <p>font-weight: bold</p> <p>color: #FF0000</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
buttonstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
stylevariant	<p>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.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	Optional	<p>VAR1</p> <p>VAR2</p>
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	<p>left</p> <p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes 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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
colspan	<p>Column spanning of control.</p> <p>If you use TR table rows then you may sometimes want to control the number of columns your control occuppies. By default it is "1" - but you may want to define the control to span over more than one columns.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	Row spanning of control.	Optional	1

	<p>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.</p> <p>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.</p>		2 3 4 5 50 int-value
imagedisabled	<p>URL of image that is displayed if the control is disabled. Use properties VISIBLEPROP and INVISIBLEMODE to disable the control.</p>	Optional	gif jpg jpeg
submitbutton	<p>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.</p> <p>i.e. password and username or complete search forms</p> <p>Default value is false.</p> <p>You should only use a 'Submitbutton' if the withformtag option of the pagebody tag is set true.</p>	Optional	true false
tabindex	<p>Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.</p>	Optional	-1 0 1 2 5 10 32767
Binding			
method	(already explained above)		
visibleprop	<p>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.</p>	Optional	
nameprop	<p>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</p>	Optional	

	<p>"nameprop" you can dynamically set the button's text by your application. Use the nameprop in cases the button's text should change dependent on your logic.</p> <p>Example: you may want to define the button's text to reflect the next status the user can set to a business object.</p>		
titleprop	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther user moves the mouse onto the control.	Optional	
Online help			
title	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	Optional	
titletextid	Text ID that is passed to the multi lanaguage 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	

47

BUTTONLIST

▪ Adapter Interface	338
▪ Properties	338

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

```
DEFINE DATA PARAMETER
1 BUTTONLIST (1:*)
2 ID (U) DYNAMIC
2 IMAGEURL (U) DYNAMIC
2 METHOD (U) DYNAMIC
2 STYLE (U) DYNAMIC
2 TEXT (U) DYNAMIC
END-DEFINE
```

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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>Sometimes it is useful to have a look into the generated HTML code in order to know where direct style definitions</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

	are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
imageheight	Pixel height of image inside button.	Optional	
imagewidth	Pixel width of image inside button.	Optional	
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	

48

CHECKBOX

■ Properties	342
--------------------	-----

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	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	<p>Horizontal alignment of control in its column.</p> <p>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.</p>	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 snychronized 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 two span over more than one columns. The property only makes sense in table rows that are snychronized 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) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	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).	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Use DISPLAYPROP to dynamically define whether the field is displayonly.	Optional	
flush	Flushing behaviour of the input control.	Optional	screen server

	<p>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.</p> <p>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.</p> <p>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 representaion directly after changing the value.</p>		
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 lanaguage 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	

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

49 COMBODYN2

▪ Adapter Interface	348
▪ Properties	348

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

```
DEFINE DATA PARAMETER
1 COSTCENTER (U) DYNAMIC
1 VALIDCOSTCENTERS (1:*)
2 ID (U) DYNAMIC
2 NAME (U) DYNAMIC
2 SELECTED (L)
END-DEFINE
```

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	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	
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 two 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
renderasfield	<p>If set to "true" then the combo box is rendered like a FIELD control that offers valid value support.</p> <p>Default is "false".</p> <p>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.</p>	Optional	true false
allowmultiselection	If set to true then multiple selections are allowed.	Optional	true false
combostyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	
invisiblemode	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	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)		
validvaluesprop	(already explained above)		
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly(true) or not (false).	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Use DISPLAYPROP to dynamically define whether the field is displayonly.	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	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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.</p>	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	

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	

50

COMBOFIX

■ COMBOFIX Properties	354
■ COMBOOPTION Properties	357

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 popup 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	
align	Horizontal alignment of control in its column.	Optional	left

	<p>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.</p> <p>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.</p>		center right
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	top middle bottom
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
combostyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p>	Optional	

	<p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		
invisiblemode	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	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)		
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly(true) or not (false).	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Use DISPLAYPROP to dynamically define whether the field is displayonly.	Optional	
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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</p>	Optional	screen server

	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 representaion 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	
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	

51

DATEINPUT

▪ Example	360
▪ Properties	360

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 popup in which the user can browse through a calendar. The calendar can be controlled by server side processing in the following way:

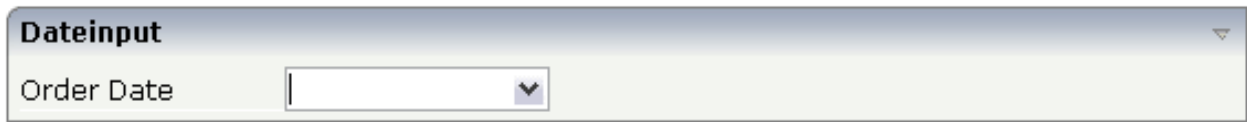
- You can define a valid-from and a valid-to date. Thus, the control will not allow the user to input an invalid date.
- You can explicitly control the color and the tooltip information inside the calendar. For example, you may set up a calendar in which vacation times are highlighted in a certain way.

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

	<p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>		180 200 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. Use DISPLAYPROP to dynamically define whether the field is displayonly.	Optional	
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p>	Optional	screen server

	<p>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.</p> <p>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.</p>		
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	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	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	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>If you want to directly control the alignment of text: in most text based controls there is an explicit</p>	Optional	left center right

	property "textalign" in which you align the control's contained text.		
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
inputstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
colspan	<p>Column spanning of control.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p>

	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
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	<p>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.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	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

popuonalt40	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 lanaguage 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 ther 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: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: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: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.	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: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 for a generated statusprop variable to which field the statusprop belongs.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control. 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 control.	Optional	

52

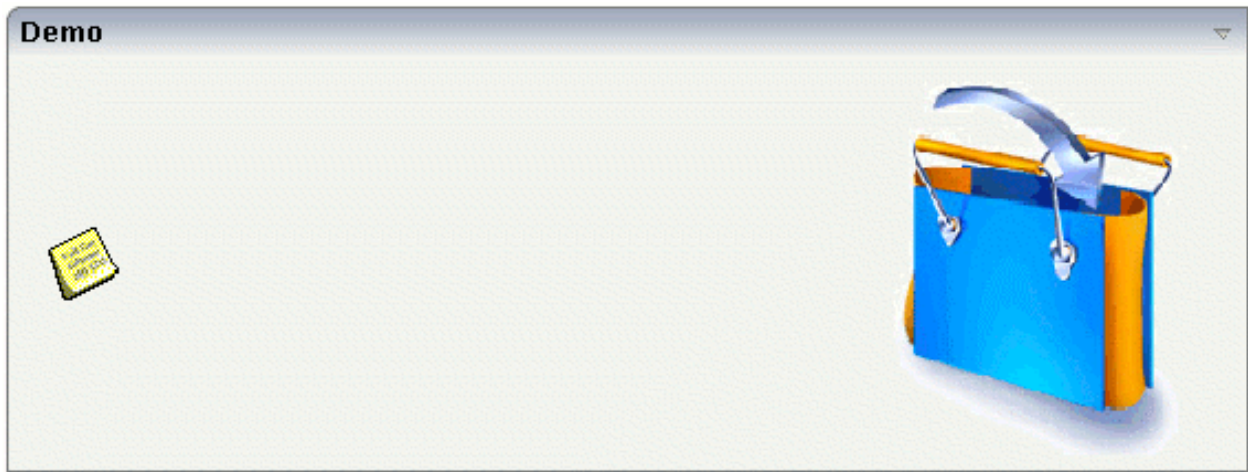
DROPICON

▪ Example	368
▪ Properties	368

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.	Obligatory	gif jpg jpeg

	"../HTMLBasedGUI/images/new.gif" point to a URL that is located inside a different project.		
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	
draginfo	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	
dropinfo	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			
draginfo	(already explained above)		
dropinfo	(already explained above)		
dropmethod	(already explained above)		
image	Name of adapter parameter that provides as value the URL of the image that is shown inside the control.	Optional	
method	(already explained above)		
visible	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	
title	Name of the adapter parameter that dynamically defines the title of the control. The title is displayed as tool tip when ther 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":	Optional	invisible cleared

	<p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>		
imageinactive	<p>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".</p> <p>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.</p> <p>If you do not define a value for this property then the icon is made invisible.</p>	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	<p>If set to "true" then 2 pixels of distance are kept on the left and on the right of the icon.</p> <p>Reason being: 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".</p>	Optional	<p>true</p> <p>false</p>
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	<p>left</p> <p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>

colstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
spanstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
tabindex	<p>Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.</p>	Optional	<p>-1</p> <p>0</p> <p>1</p> <p>2</p> <p>5</p> <p>10</p> <p>32767</p>
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 lanaguage management - representing the tooltip text that is used for the control.	Optional	
titleprop	(already explained above)		

53

FIELD

■ Built-in Events	374
■ Properties	374

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

- Normal input/output of text.
- Password input.
- 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 popup 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 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)		
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 into this FIELD. 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
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
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	Optional	left center right

	<p>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.</p> <p>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.</p>		
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
fieldstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>

	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.		
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 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.	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) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	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)		
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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 representaion directly after changing the value.</p>	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	
displayprop	Name of the adapter parameter that dynamically passes information whether the field is displayonly(true) or not (false).	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Use DISPLAYPROP to dynamically define whether the field is displayonly.	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	Optional	

	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.		
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
maxlengthprop	Name of the adapter parameter that provides the maximum number of characters that a user may enter into this FIELD. 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 from 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

	<p>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.</p> <p>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.</p>		xs:dateTime xs:time ----- N n.n P n.n string n xs:byte xs:short
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 way to input the value. The hint is not language dependent.	Optional	
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 specifies how many digits are to be displayed (ie 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	Number that specifies how many decimal digits are to be displayed. 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. See at chapter 'Popup Dialog Management' for more details. 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 openIdValueComboOrPopup
popupinputonly	Boolean property that control if a field with POPUPMETHOD defined is still usable for keyboard input. If "false" (= default) then the	Optional	true false

	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.		
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 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. Specifiying "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	Optional	true false

	consequence you can use this control for making inputs through a touch terminal.		
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 lanaguage 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	Comma 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 comma 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: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: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: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.	Optional	

	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.		
njx:natsysvar	If the control shall be bound to a Natural system variable, this attribute specifies the name of the system variable.	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 for a generated statusprop variable to which field the statusprop belongs.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control. 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 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	
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

54

FILEUPLOAD/FILEUPLOAD2

■ FILEUPLOAD	386
■ FILEUPLOAD2	388
■ FILEUPLOAD Properties	389
■ FILEUPLOAD2 Properties	392

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.

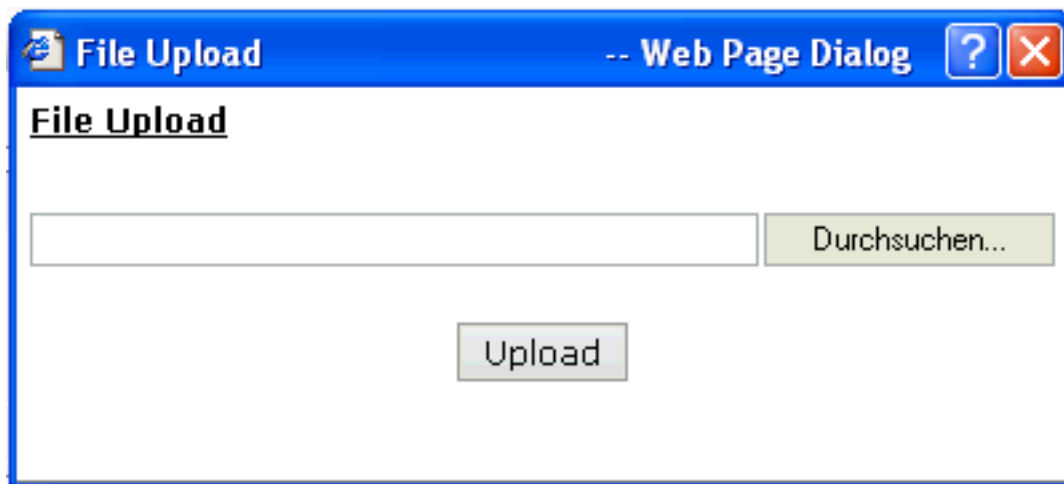
FILEUPLOAD

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



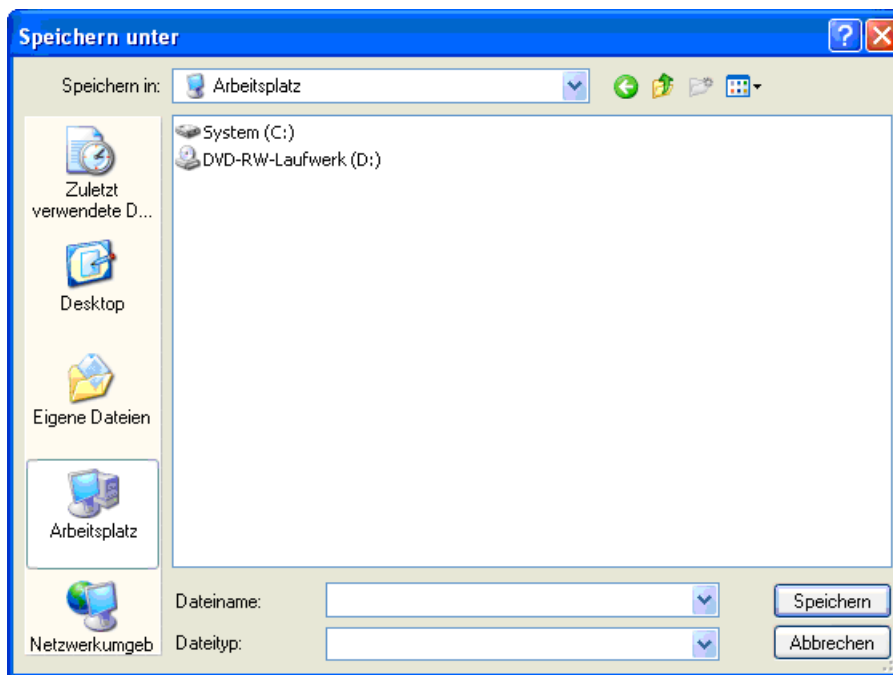
The screenshot shows a web form titled "File upload" in a light blue header. Below the header, there is a button labeled "Upload File ...". Underneath the button, there are two input fields: "Client file name" and "Server file name", each followed by a text input box.

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

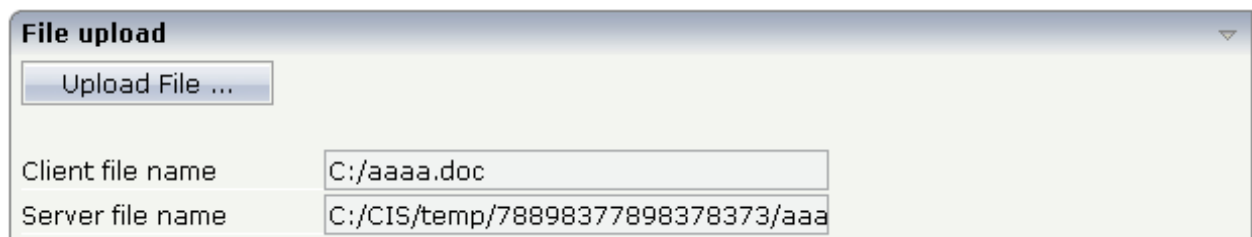


The screenshot shows a standard Windows-style dialog box titled "File Upload" with a blue title bar. The title bar also contains the text "-- Web Page Dialog" and standard help and close buttons. The main area of the dialog has a title "File Upload" and a text input field. To the right of the input field is a button labeled "Durchsuchen...". Below the input field and button is a large "Upload" button.

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:

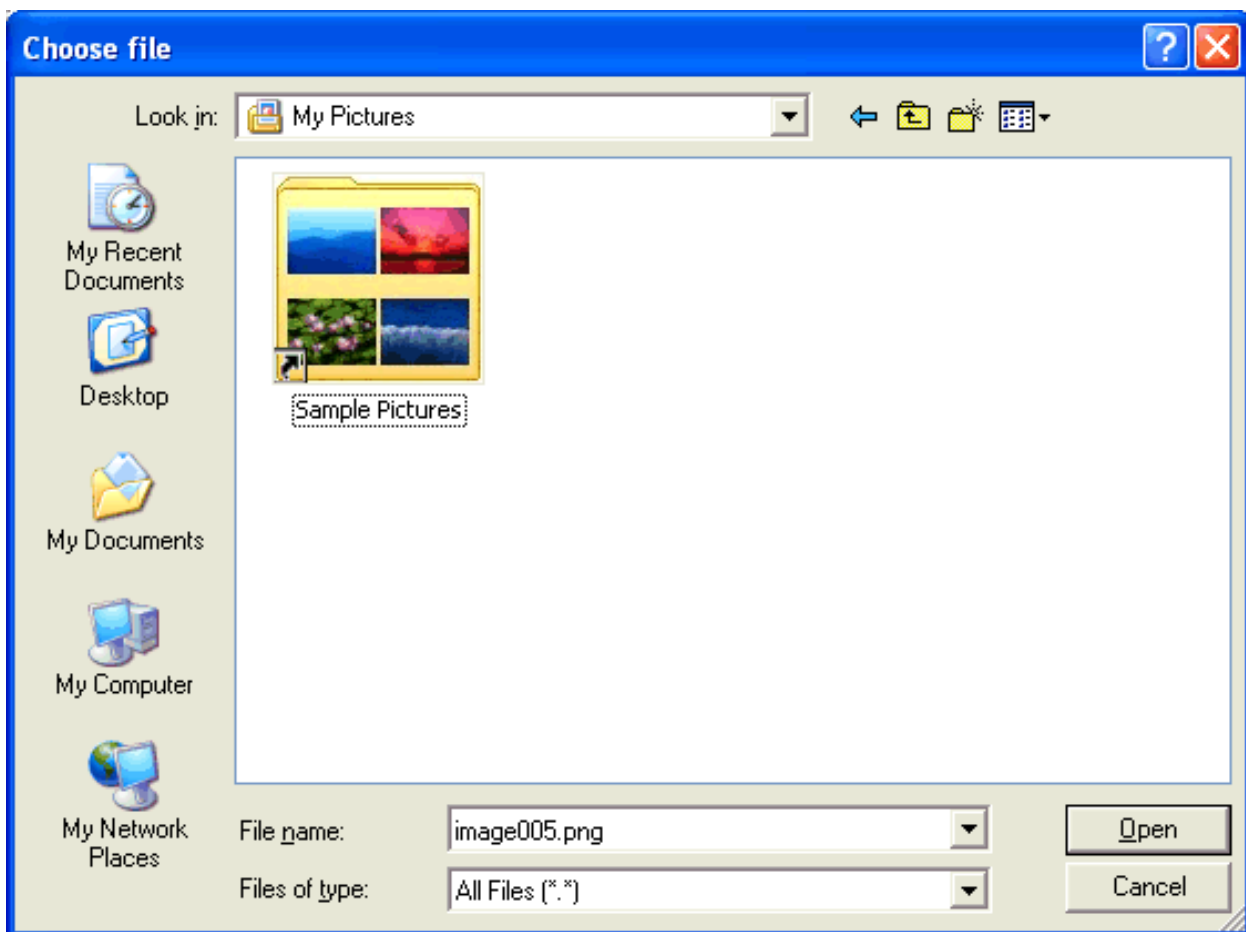


FILEUPLOAD2

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	
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.	Optional	100

	<p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>		120 140 160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>This property has three possible values:</p> <p>(1) "invisible": the button is not visible without occupying any space.</p> <p>(2) "disabled": the button is deactivated: it is "grayed" and does not show any roll over effects any more.</p> <p>(3) "cleared": the button is not visible but it still occupies space.</p>	Optional	invisible cleared
buttonstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p>	Optional	

	<p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	<p>left</p> <p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>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 specifies the position of the control inside the column.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p>

	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
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	Optional	

	server file system. Note that this file name is not the same as the client file name.		
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	
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".	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	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	Optional	invisible disabled cleared
Appearance			
invisiblemode	(already explained above)		
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value

colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
darkbackground	<p>Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used.</p> <p>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.</p>	Optional	true false

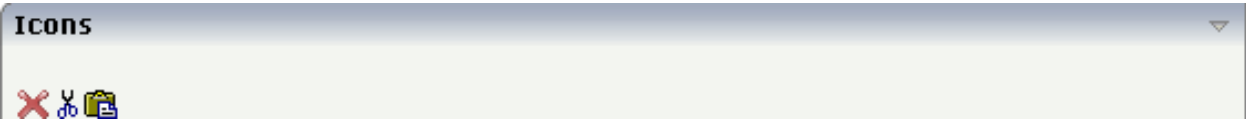
55

ICON

▪ Example	396
▪ Properties	396

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	Optional	gif jpg jpeg

	<p>generated page is located directly inside your project's directory.</p> <p>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.</p>		
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	<p>Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime.</p> <p>Do not specify a "name" inside the control if specifying a "textid".</p>	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	
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	
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	<p>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".</p> <p>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</p>	Optional	gif jpg jpeg

	<p>icon image. Maybe the image is a grayed version of the normal icon image.</p> <p>If you do not define a value for this property then the icon is made invisible.</p>		
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	<p>left</p> <p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
withdistance	<p>If set to "true" then 2 pixels of distance are kept on the left and on the right of the icon.</p> <p>Reason being: 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".</p>	Optional	<p>true</p> <p>false</p>
colstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>Sometimes it is useful to have a look into the generated HTML code in order to know where direct</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>

	style definitions are applied. Press right mouse-button in your browser and select the "View source" or "View frame's source" function.		
spanstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
invisiblemode	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	Optional	<p>invisible</p> <p>cleared</p>
tabindex	<p>Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.</p>	Optional	<p>-1</p> <p>0</p> <p>1</p> <p>2</p> <p>5</p> <p>10</p> <p>32767</p>
nameposition	<p>Position of the (optional) text to the icon. Aside or below, default is aside.</p> <p>Set the corresponding text in the name or the text id property.</p>	Optional	<p>aside</p> <p>below</p>

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 ther user moves the mouse onto 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 lanaguage 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	

56

ICONLIST

■ Adapter Interface	402
■ Built-in Events	402
■ Properties	402

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

```
DEFINE DATA PARAMETER
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
END-DEFINE
```

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 arranged in one row, one aside the other.	Optional	true false
cellspacing	An icon of the ICONLIST control is embedded into an internal cell. The CELLSPACING property defines the number of pixels that are kept between the icon and 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	
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	
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	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

57 IHTML

■ Properties	406
--------------------	-----

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p> <p>50%</p> <p>100%</p>
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	<p>100</p> <p>150</p> <p>200</p> <p>250</p> <p>300</p> <p>250</p> <p>400</p> <p>50%</p> <p>100%</p>

colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
htmlstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	Optional	
valign	<p>Vertical alignment of control in its column.</p> <p>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</p>	Optional	top middle bottom

	bigger than the size of the control. In this case the "align" property specify the position of the control inside the column.		
--	---	--	--

58

IMAGEOUT

■ Properties	410
--------------------	-----

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p> <p>50%</p> <p>100%</p>
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	
colspan	Column spanning of control.	Optional	

	<p>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.</p> <p>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.</p>		
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	Optional	

59

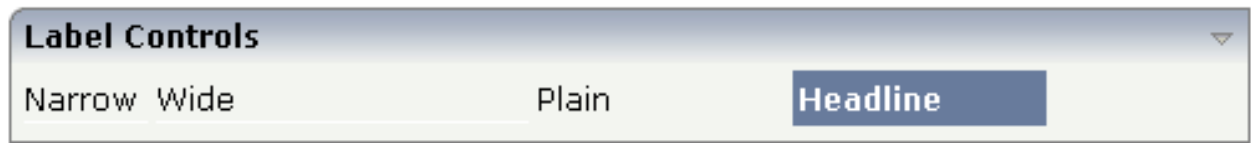
LABEL

■ Example	415
■ Aligning the Text	415
■ Properties	416

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%
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. You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.	Optional	true false
width	(already explained above)		
height	Height of the control. There are three possibilities to define the height:	Optional	100 150 200

	<p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>		<p>250</p> <p>300</p> <p>250</p> <p>400</p> <p>50%</p> <p>100%</p>
asheadline	<p>If set to true, the label has a dark background and the text is written in white (if using the standard style sheet).</p> <p>You may use this rendering style is you use labels as headlines of control grids (ROWTABLEAREA2 control).</p>	Optional	<p>true</p> <p>false</p>
asplaintext	<p>If set to true, no white line is drawn under the label text (if using the standard style sheet).</p> <p>You may use this rendering style if the label is used to name a RADIOBUTTON control or a CHECKBOX control.</p>	Optional	<p>true</p> <p>false</p>
textalign	Horizontal alignment of the text that is shown.	Optional	<p>left</p> <p>center</p> <p>right</p>
cuttext	<p>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.</p> <p>Default is "false".</p>	Optional	<p>true</p> <p>false</p>
labelstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>

	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.		
labelstyleclass	CSS style class used for rendering.	Optional	
stylevariant	<p>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.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	Optional	VAR1 VAR2 VAR3 VAR4
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	left center right
valign	<p>Vertical alignment of control in its column.</p> <p>Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes 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.</p>	Optional	top middle bottom
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	1 2 3 4 5 50 int-value

rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
invisiblemode	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	Optional	<p>invisible</p> <p>cleared</p>
Binding			
visibleprop	<p>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.</p>	Optional	
Online Help			
title	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	Optional	
titletextid	<p>Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.</p>	Optional	

60

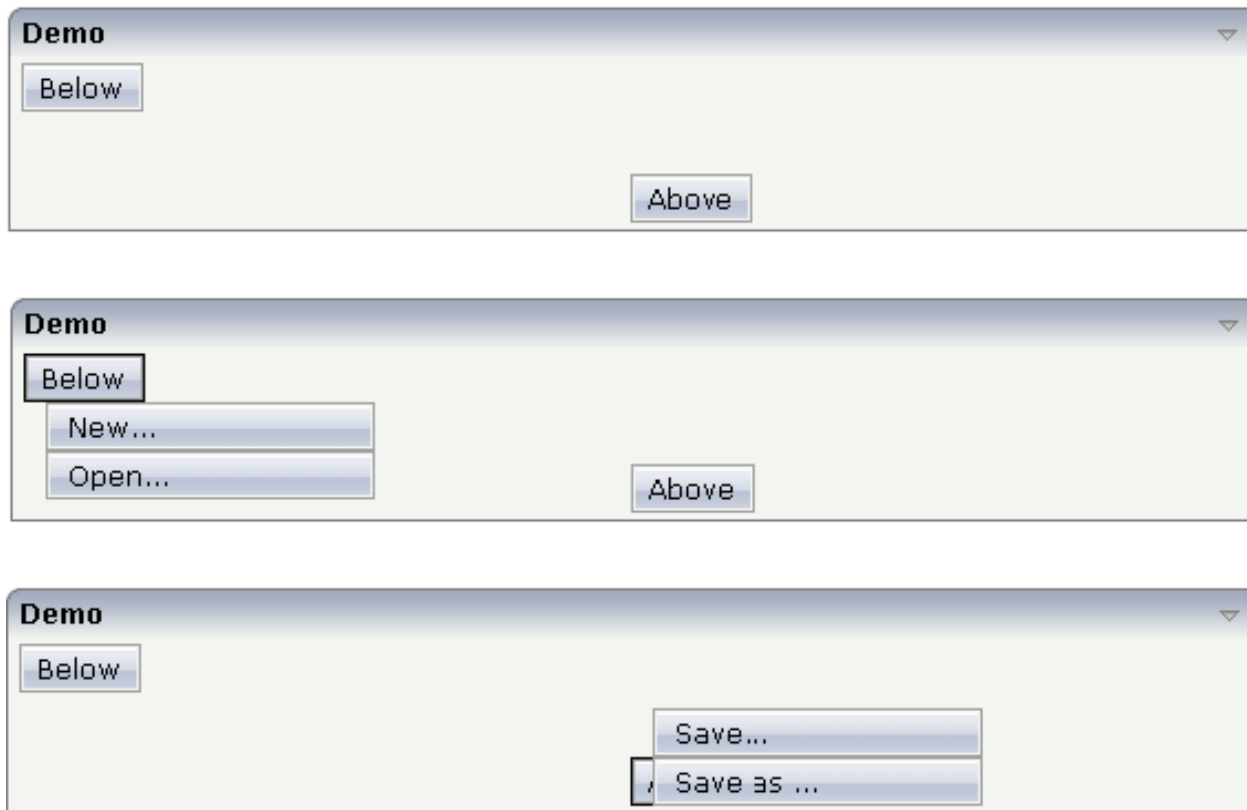
MENUBUTTON

■ Example	422
■ MENUBUTTON Properties	423
■ MENUITEM Properties	425

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	Optional	100 120 140 160 180 200 50%

	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.		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	
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 two 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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	

61

METHODLINK

■ Properties	428
--------------------	-----

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	<p>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.</p> <p>Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".</p> <p>MOZILLA: this property is not available in Mozilla!</p>	Optional	<p>true</p> <p>false</p>
linkstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
linkclass	<p>CSS style class definition that is directly passed into this control.</p> <p>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 ADDSTYLE SHEET property of the PAGE tag.</p>	Optional	
align	<p>Horizontal alignment of control in its column.</p> <p>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</p>	Optional	<p>left</p> <p>center</p> <p>right</p>

	<p>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.</p> <p>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.</p>		
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
nowrap	<p>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.</p> <p>You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.</p>	Optional	<p>true</p> <p>false</p>
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>The property only makes sense in table rows that are synchronized within one container (i.e. TR, STR</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p>

	table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.		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	

62

MULTISELECT

■ Example	434
■ Adapter Interface	434
■ Properties	434

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

Example

Sevilla		Lebrija
Carmona		Malaga
Cadiz	>>	Bilbao
Valencia		
Madrid	>	
Salamanca		
Barcelona	<	
Granada	<<	

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

```
DEFINE DATA PARAMETER
1 TOWNS (1:*)
2 ID (U) DYNAMIC
2 SELECTED (L)
2 TEXT (U) DYNAMIC
END-DEFINE
```

Properties

Basic			
valueprop	Name of the adapter parameter representing this control in the application.	Obligatory	
width	Width of the control.	Obligatory	100
	There are three possibilities to define the width:		120
			140

	<p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>		160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>Horizontal alignment of control in its column.</p> <p>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</p>	Optional	left center right

	<p>column around the controls exactly is sized in the same way as the contained control.</p> <p>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.</p>		
valign	<p>Vertical alignment of control in its column.</p> <p>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 specifies the position of the control inside the column.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
msstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p>	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.		
Binding			
valueprop	(already explained above)		
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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.</p>	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	

63

NEWSFEED

■ Example	441
■ Built-in Events	442
■ Properties	442

The NEWSFEED control is a simple-to-use “newsreader” within the Application Designer pages. It offers the possibility to read news feeds (RSS feeds and Atom feeds).



Important: In order to use the NEWSFEED control, you have to specify a valid RSS or Atom feed URL (for example http://news.cnet.com/2547-1001_3-0-5.xml). If necessary, you also have to specify your proxy server settings (host, port, user name, password).

Example



The XML layout definition is:

```
<rowarea name="Newsfeed Control" width="560">
  <newsfeed infoprop="newsfeedinfoprop" width="550" height="450">
    </newsfeed>
  </rowarea>
```

Built-in Events

value-of-infoprop.onOpenLink
value-of-infoprop.onOpenLinkNewTarget

Properties

Basic			
infoprop	Name of the adapter parameter that represents the control in the adapter.	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 percantage 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%
splitstyle	By default the newsfeed control appears within a vsplit control. Headers on the left and content on the right. Set this value to hsplit and the control appears within a hsplit control. Headers on top, content on the bottom.	Optional	vsplit
			hsplit
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

64

RADIOBUTTON

■ Properties	444
--------------------	-----

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

	<p>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.</p> <p>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.</p>		<p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
invisiblemode	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p>	Optional	<p>invisible</p> <p>cleared</p>

	(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.		
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).	Optional	

statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should act. Use DISPLAYPROP to dynamically define whether the field is displayonly.	Optional	
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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.</p>	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	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	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	

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

65

SCHEDULELINE

■ Properties	450
--------------------	-----

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	<p>Name of the adapter parameter that represents the control in the adapter.</p> <p>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.</p> <p>Example: #FF0000\ "1000;Text 1;1;#00FF00;500;Text 2;2</p> <p>The total "logical width" is 1500. The first item occupies 2/3 of the width, the right item occupies 1/3 of the width.</p> <p>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.</p>	Obligatory	

width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Obligatory	100 120 140 160 180 200 50% 100%
pixelheight	Height of the control in pixels.	Optional	
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	Optional	
Appearance			
width	(already explained above)		
pixelheight	(already explained above)		
pixelsizemode	<p>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.</p> <p>When switching this property to "true" then the size of the sections are interpreted as real pixel values.</p>	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 two 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

	<p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		<p>color: #0000FF</p> <p>font-weight: bold</p>
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	<p>Name of an adapter parameter that is used in the following way:</p> <p>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.</p>	Optional	
preselectmode	<p>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.</p> <p>Default is "false".</p> <p>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.</p>	Optional	<p>true</p> <p>false</p>
Vertical			
verticalschedule	Flag that indicates if the line is rendered vertically. Default is false.	Optional	<p>true</p> <p>false</p>

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
dropinfo	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 DROPINFO to get the id of the dragged cell (source). Use SELSCHEDULE 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	

66 SLIDER

■ Example	456
■ Adapter Interface	457
■ Properties	457

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.

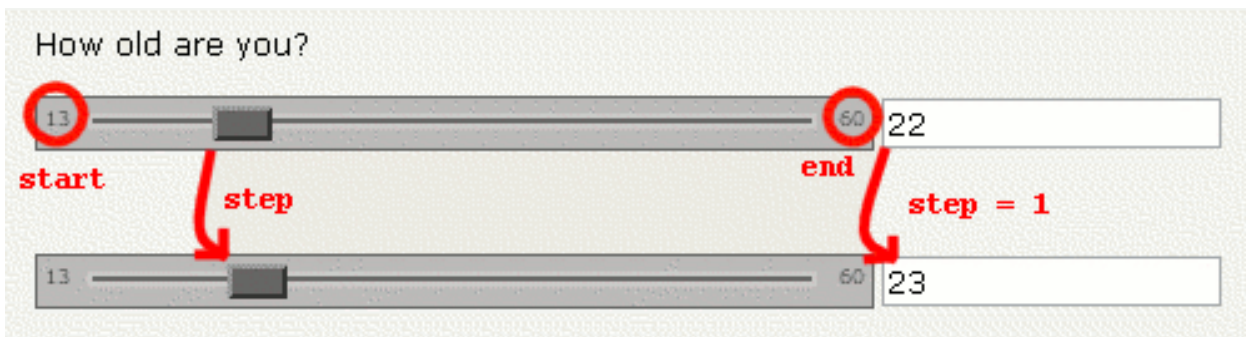
Example



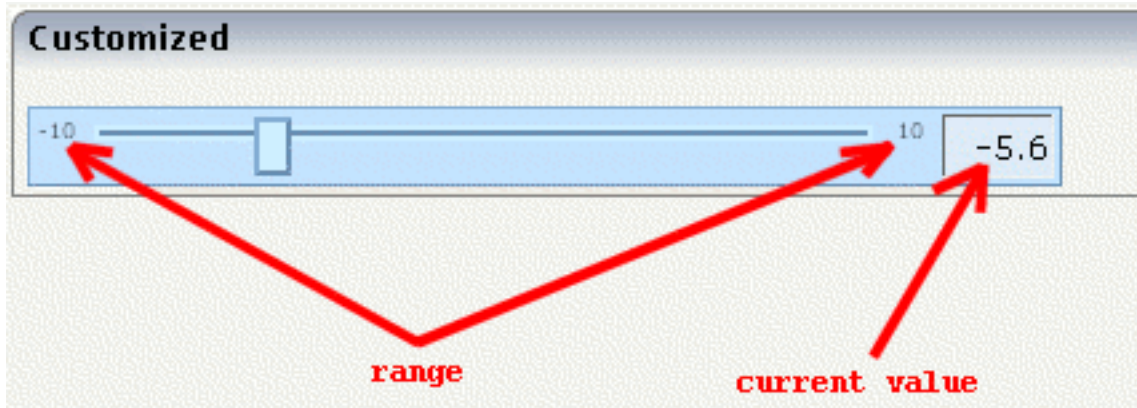
The XML layout definition is:

```
<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

```

DEFINE DATA PARAMETER
1 SLIDER
2 DISPLAYONLY (L)
2 FROM (F4)
2 SLIDERVALUE (F4)
2 STEP (F4)
2 TO (F4)
END-DEFINE

```

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. 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
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. 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
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.	Optional	#FF0000 #00FF00 #0000FF

	This should be a valid CSS color value. For example a name(blue, red), a hexadecimal value(#99CCFF) or others.		#FFFFFF #808080 #000000
valuebordercolor	Background color of the slider current value which is shown if the "showcurrentvalue" property is set to true. 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	Optional	#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%

67

STRIPSEL

▪ Example	464
▪ Properties	464

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.	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.		
scrollable	Flag that indicates if the control shows scroll icons on the right upper corner. Default is true	Optional	true false
backgroundstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	<p>Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime.</p> <p>Do not specify a "name" inside the control if specifying a "textid".</p>	Optional	
scrollrighttitle	Help text that is displayed if the user moves the mouse of the scroll to right icon.	Optional	
scrollrighttitletextid	<p>Multi language dependent text that is displayed inside the control. The "textid" is translated into a corresponding string at runtime.</p> <p>Do not specify a "name" inside the control if specifying a "textid".</p>	Optional	
scrollleftimage	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p>	Optional	gif jpg jpeg

	<p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>		
scrollleftimagertl	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	gif jpg jpeg
scrollrightimage	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	gif jpg jpeg
scrollrightimagertl	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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</p>	Optional	gif jpg jpeg

	<p>"../HTMLBasedGUI/images/new.gif" will point to an image of a neighbour project.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>		
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

68

SUBPAGE

■ Properties	470
--------------------	-----

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	<p>Name of the adapter parameter that provides the URL to be displayed inside the SUBPAGE control.</p> <p>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.</p>	Obligatory	
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Sometimes obligatory	100 120 140 160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p>	Sometimes obligatory	100 150 200 250 300

	<p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>		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	<p>Definition of the scrollbar's appearance.</p> <p>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").</p> <p>Default is "auto".</p>	Optional	auto yes no
pagestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value

rowspan	<p>Row spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
Binding			
valueprop	(already explained above)		

69

TABSEL

■ Adapter Interface	474
■ Built-in Events	475
■ Properties	475

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

```
DEFINE DATA PARAMETER
1 TABS
2 SELECTEDITEM (I4)
2 TSITEMS (1:*)
3 ID (U) DYNAMIC
3 NAME (U) DYNAMIC
3 TITLE (U) DYNAMIC
END-DEFINE
```


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	

70

TABSTRIP2

▪ Example	478
▪ Adapter Interface	478
▪ Built-in Events	478
▪ Properties	479

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

```
DEFINE DATA PARAMETER
1 TABS
2 SELINDEX (I4)
2 TSITEMS (1:*)
3 NAME (U) DYNAMIC
END-DEFINE
```

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.	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	

71 TAGCLOUD

▪ Example	482
▪ Adapter Interface	483
▪ Built-in Events	483
▪ Properties	483

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

```
DEFINE DATA PARAMETER
1 TAGCLOUD
2 TCLITEM (1:*)
3 ID (U) DYNAMIC
3 POPULARITY (I4)
3 TEXT (U) DYNAMIC
END-DEFINE
```

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").	Optional	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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 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

72

TEXT

■ Properties	488
--------------------	-----

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

Properties

Basic			
valueprop	Name of the adapter parameter that provides the content of the control.	Obligatory	
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Sometimes obligatory	100 120 140 160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 350 400 450 50% 100%
flush	<p>Flushing behaviour of the input control.</p> <p>By default an input into the control is registered within the browser client - and communicated to the server adapter object when a user</p>	Optional	screen server

	<p>e.g. presses a button. By using the FLUSH property you can change this behaviour.</p> <p>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.</p> <p>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.</p>		
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	<p>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).</p> <p>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.</p>	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)		
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).	Optional	
statusprop	Name of the adapter parameter that dynamically passes information how the control should be rendered and how it should	Optional	

	act. Use DISPLAYPROP to dynamically define whether the field is displayonly.		
wrap	<p>Specifies the line wrapping inside the control. By default a line that exceeds the width of the control is broken automatically.</p> <p>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.</p> <p>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 carefule when specifying "hard" as consequence!</p>	Optional	soft hard off
rows	<p>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!</p> <p>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).</p>	Optional	
cols	<p>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!</p> <p>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).</p>	Optional	
colspan	<p>Column spanning of control.</p> <p>If you use TR table rows then you may sometimes want to control the number of columns your control occupys. By default it is "1" - but you may want to define the control to span over more than one columns.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	1 2 3 4 5 50 int-value
maxlength	<p>Maximum number of characters that a user may enter into this FIELD. 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.</p>	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 into this FIELD. Consider to use MAXLENGTH to define this number in a static way.	Optional	
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
textareastyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	
title	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	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	
scroll	<p>Definition of the scrollbar's appearance.</p> <p>You can define that the scrollbars only are shown if the content is exceeding the control's area ("auto"). Or scrollbars can be shown</p>	Optional	auto scroll hidden

	always ("scroll"). Or scrollbars are never shown - and the content is cut ("hidden"). Default is "auto".		
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			
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: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: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: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.	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: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 for a generated statusprop variable to which field the statusprop belongs.	Optional	

njx:natcv	Name of a Natural control variable that shall be assigned to the control. 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 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	

73

TEXTOUT

▪ Example	496
▪ Properties	496

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	Sometimes obligatory	100 120 140 160 180 200 50% 100%

	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.		
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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
nowrap	<p>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.</p> <p>You can avoid this behaviour by setting NOWRAP to "true". No line break will be performed by the browser.</p>	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	<p>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).</p> <p>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.</p>	Optional	date float int long time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time ----- N n.n P n.n string n xs:byte xs:short
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

	<p>Example: if you want to output the source of an HTML text then STRAIGHTTEXT should be set to "true".</p> <p>MOZILLA: this property is not available in Mozilla!</p>		
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	<p>left</p> <p>center</p> <p>right</p>
valign	<p>Vertical alignment of control in its column.</p> <p>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.</p>	Optional	<p>top</p> <p>middle</p> <p>bottom</p>
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>

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 FG_COLORPROP 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 BG_COLORPROP to choose both - the text and background color.	Optional	
textoutstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
textoutclass	<p>CSS style class definition that is directly passed into this control.</p> <p>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 ADD_STYLE_SHEET property of the PAGE tag.</p>	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.	Optional	VAR1 VAR2 VAR3 VAR4

	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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!		
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 ther 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) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.	Optional	invisible cleared
Natural			
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: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: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.	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:natcomment	The value of this attribute is generated as comment line into the parameter data area of the Natural adapter,	Optional	

	before the field name. The Map Converter, for instance, uses this attributes to indicate for a generated statusprop variable to which field the statusprop belongs.		
njx:natcv	Name of a Natural control variable that shall be assigned to the control. 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 control.	Optional	

74 TOGGLE

■ Properties	504
--------------------	-----

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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	
partialimage	Image URL that is shown if the corresponding property value is "null".	Optional	
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	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	<p>CSS style class definition that is directly passed into this control.</p> <p>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.</p>	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. Use DISPLAYPROP to dynamically define whether the field is displayonly.	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	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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 representaion directly after changing the value.</p>	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	Optional	

	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			
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 lanaguage management - representing the tooltip text that is used for the control.	Optional	

75

ACTIVEX

■ Properties	510
--------------------	-----

This is a “hot topic”: embedding ActiveX controls in pages. Before telling you what the control does, let us explain why we do it:

Of course, the client integration of ActiveX controls has - from browser or SWT perspective - only disadvantages:

- ActiveX controls are not secure: you decide to run one control or not. But do not have a “sandbox” as you have with JavaScript or with applets. Using an ActiveX control means that this control - once running - has native access to your computer, just as any other native program.
- ActiveX controls are bound to the Microsoft Windows platform.
- ActiveX controls need to be explicitly installed on the client side - maybe automated in some way, but still an explicit installation is necessary.

But - and this is why we support them - in some cases, they are a nice way to integrate other software which runs out of the scope of the browser.

Example: you may want to integrate your user interface with a barcode reader which is connected to your client via a serial interface. In this case, there is no way to access this barcode reader via JavaScript. You need to use an ActiveX control (or a signed applet) to connect to the serial device.

There is a simple interface between HTML/JavaScript and ActiveX, and vice versa. ActiveX controls can be embedded into an HTML page and it is possible to directly access properties of the ActiveX control from JavaScript. This interface was used for building the ACTIVEX control that you can use as an Application Designer control.

Properties

Basic			
classid	Class id of the ActiveX control. A string in the format "8E27C92B-1264-101C-8A2F-040224009C02" representing the UUID of the ActiveX component. The CLASSID is used inside the HTML client to reference the ActiveX control.	Optional	
progid	The unique program identifier which has been registered for this ActiveX Control like "Shell.Explorer"	Optional	
xinitparams	Init parameters that are used for creating an instance of the ActiveX control. Values are passed as semicolon separated string: property;value;property;value etc. The property is the name of the ActiveX control's property that is initialized with the corresponding value.	Optional	
setxparams	Same as GETXPARAMS but now the other direction. Adapter properties that are transferred (on change) into corresponding ActiveX properties with each	Optional	

	response. The string format is the same: activexProperty;adapterProperty;activexProperty;adapterProperty etc.		
getxparams	<p>Semicolon separated list of which ActiveX control are linked with which adapter properties. The format is: activexProperty;adapterProperty;activexProperty;adapterProperty etc.</p> <p>With each request send from the browser the ActiveX properties are collected in from the ActiveX control and are transferred (if they have changed) into the corresponding adapter properties.activex_attr_progid"Program id of the ActiveX control. E.g. "MSCAL.Calendar" for the Microsoft calendar. The PROGID is used inside the SWT client to access the ActiveX control.</p>	Optional	
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	100 120 140 160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
reloadprop	Name of the adapter parameter that indicates that the ActiveX control is reloaded with every response from the server that changed data of the ActiveX control.	Optional	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	

76

GOOGLEMAP2

■ Before You Start	514
■ Example	515
■ Typical Problems	516
■ Properties	517

The GOOGLEMAP2 control is used to provide for Google Maps support within Application Designer pages. The control internally makes use of the Google Maps API. In order to use the control on your site, you need to sign up for a Google Maps API key at <http://code.google.com/apis/maps/signup.html>. Make sure that you agree with the Google Maps API Terms of Use (<http://code.google.com/apis/maps/terms.html>).

Before You Start

In order to use the GOOGLEMAP2 control, you need to sign up for a Google Maps API key. A key is valid for a single “directory” on your web server only, i.e. you sign up for a URL like *http://www.mysite.com/mywebapp/myproject*. With a standard installation of Application Designer on localhost, you may sign up for the URL *http://localhost:8080/mywebapp/myproject*. Typically, you develop your Application Designer web application not on the site on which you run it later in productive mode. Therefore, you may sign up for two different sites (development and production site).

Required Steps

1. Choose the project directory that keeps the layouts using the GOOGLEMAP2 control.
2. Sign up for a Google Maps API key at <http://code.google.com/apis/maps/signup.html> for this project directory (e.g. *http://localhost:8080/mywebapp/myproject*).
3. Create the API key page. Store the key page in the registered project directory. You are free in naming the file (the file extension must be “html”). The GOOGLEMAP2 control embeds your API key as a subpage. The subpage must have the following minimum structure:

```
<html>
  <head>
    <script src="
http://maps.google.com/maps?file=api&v=2.x&key=YOUR_API_KEY"></script>
    <script src="../HTMLBasedGUI/general/googlemapscript.js"></script>
  </head>
  <body>
    <div id="map" style="position:absolute; top:0; left:0;"></div>
  </body>
</html>
```

You see that the page includes two JavaScript libraries. The first line refers to the Google Maps API. Replace the placeholder “YOUR_API_KEY” with your Google Maps API key. With the second line, the page includes the control’s scripting (calls from Application Designer to the Google Maps). The page body is quite simple: it contains a single `div` tag with the ID “map”. This `div` is used as an anchor to insert Google Maps controls dynamically.

Example

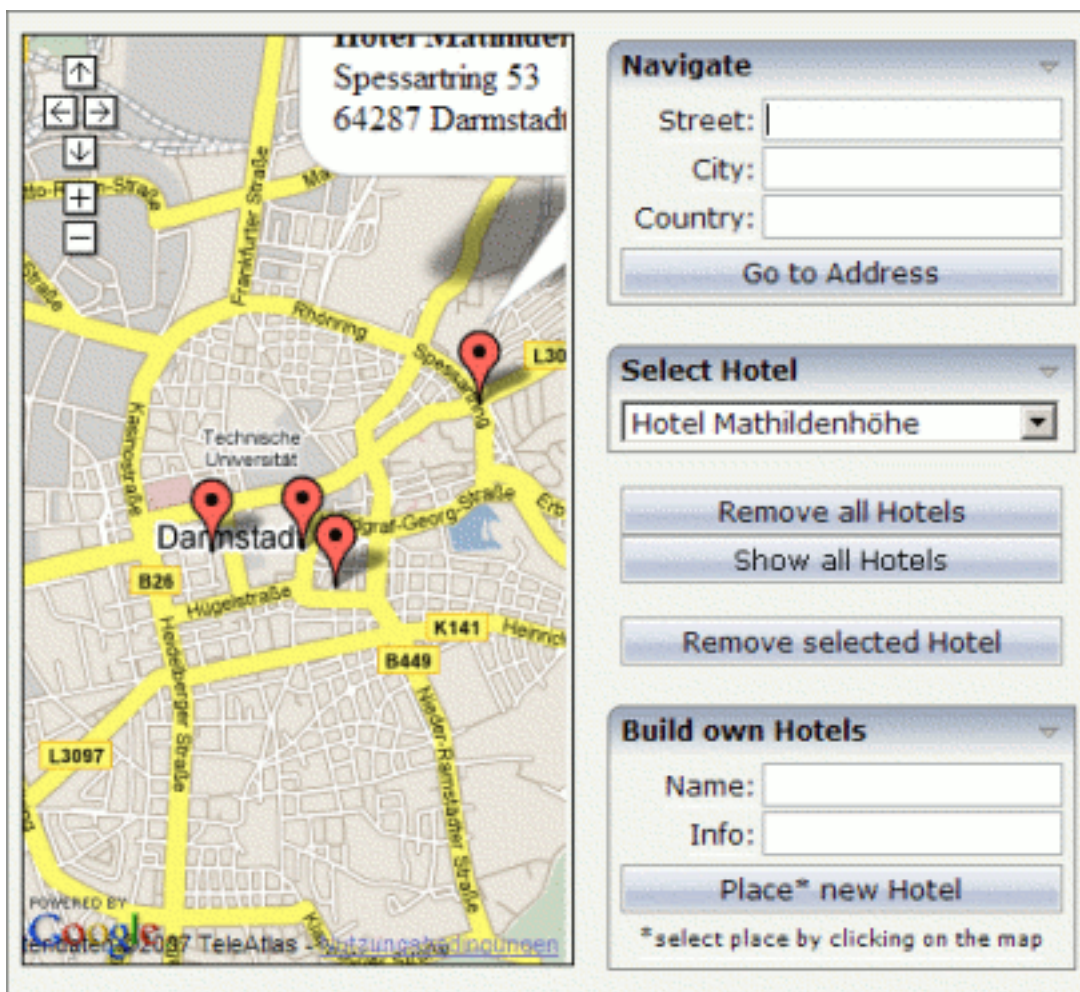
■ General Usage

General Usage

The map options are taken from the property `infoprop`. On this object, you may set the address (or latitude and longitude), the zoom level and the map size as well as the map type.



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

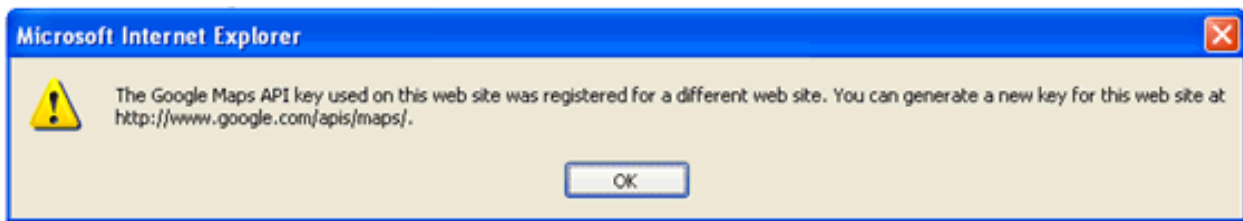


Typical Problems

- Google Map API Key
- Map Remains Gray

Google Map API Key

Your Google Maps API key is bound to a directory on a certain web server (i.e. you sign up for the URL `http://mycomputer.mydomain.com:8080/mywebapp/myproject`). If you use your key for another URL, Google shows an error message:



Reasons that cause the error:

- You have registered your computer using the computer's name (e.g. `http://mycomputer...`). But the Application Designer development workplace is started using the URL `http://localhost...`

Solution: start the Application Designer workplace with `http://mycomputer...`

- The registered directory (e.g. `.../mywebapp/myproject`) does not match your installation (either a mistake in writing when signing up for the key or you have renamed the web application or project after registration).

Solution: rename your web application or project to match the registered names. Or sign up for a new key and insert the new key into the API key page. In the latter case, delete the content of the browser's cache. Otherwise, the browser will use the former API key page (and thus the old key).

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			
infoprop	Name of adapter parameter that represents the control in the adapter.	Obligatory	
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	Obligatory	
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	100 120 140 160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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			
pagestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value

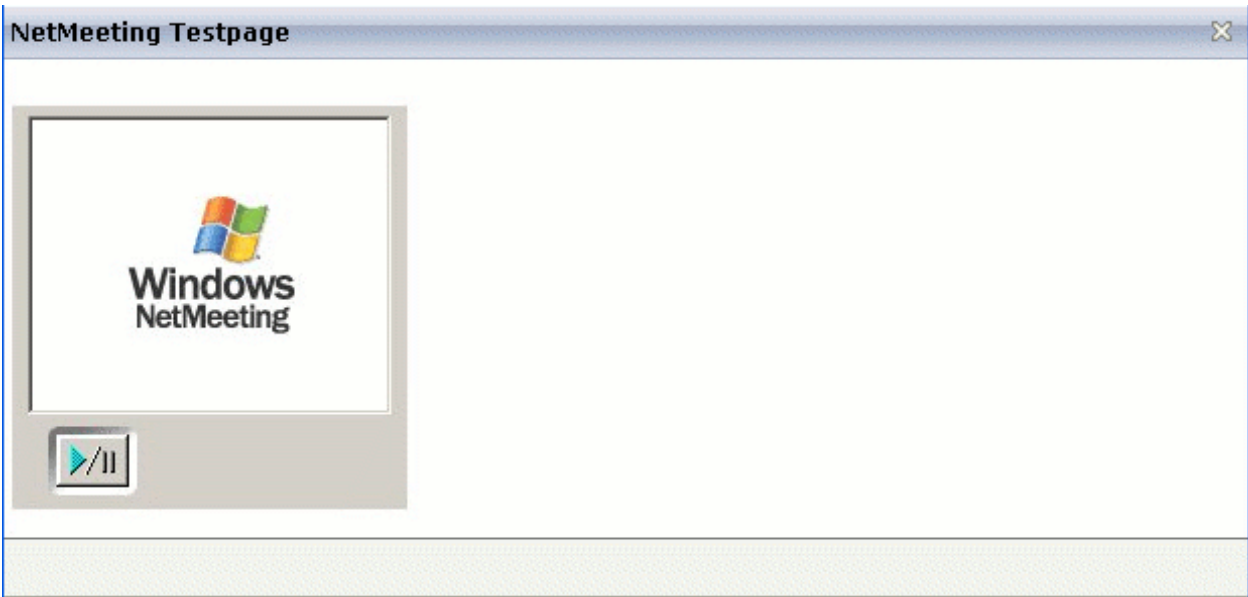
77

NETMEETING

▪ Example	520
▪ Properties	520

The NETMEETING control allows you to start NetMeeting sessions within your Application Designer pages.

Example



The XML layout definition is:

```
<pagebody>
  <itr>
    <netmeeting calltoprop="callto" modeprop="modep" width="300">
    </netmeeting>
  </itr>
</pagebody>
```

Properties

Basic			
calltoprop	Name of the adapter parameter that provides the contact data of the 'contact' that should be called. The data has to have the following semantics. ILS Server/email adress e.g. ils.netmeeting.de/contact@testmail.com	Optional	
modeprop	Name of the adapter parameter that holds the mode of the control.	Optional	

	<p>Possible are:</p> <p>FULL, PREVIEWONLY, PREVIEWNOPAUSE, REMOTEONLY, REMOTENOPAUSE, DATAONLY</p>		
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	100 120 140 160 180 200 50% 100%

78

SKYPECALL

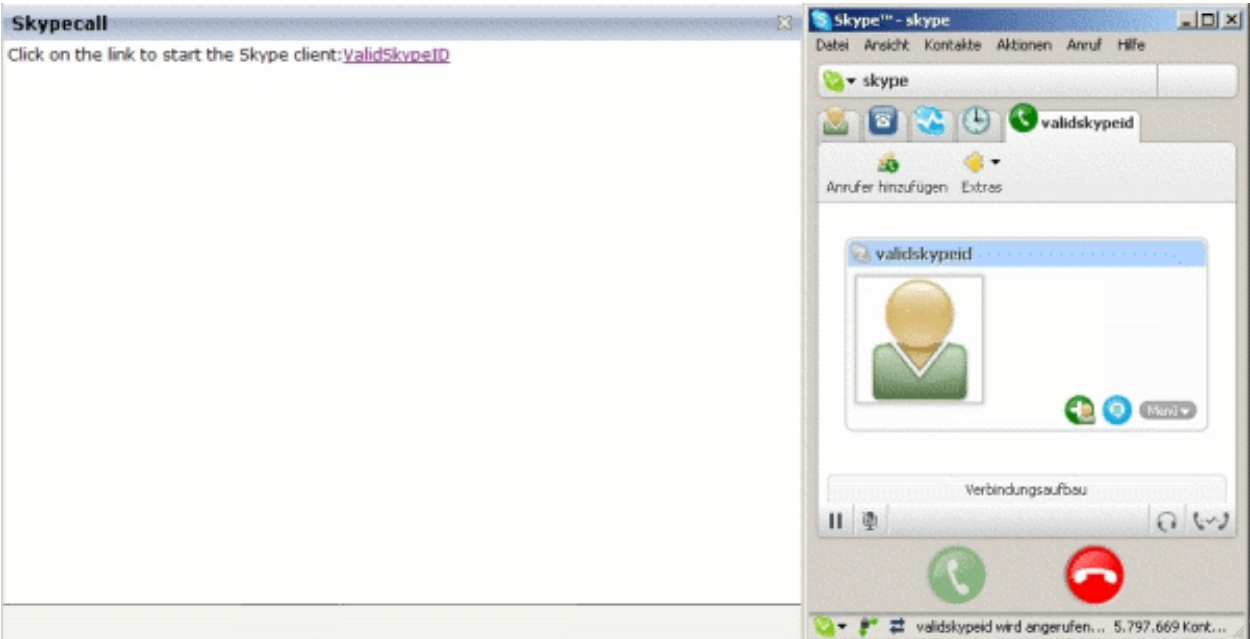
▪ Example	525
▪ Properties	525

The SKYPECALL control allows you to start the Skype client with given contact data from your Application Designer pages.



Important: In order to use the SKYPECALL control you need to have a valid Skype account and the Skype client must be installed. For further information, see <http://www.skype.com/>.

Example



The XML layout definition is:

```
<pagebody>
  <itr>
    <label name="Click on the link to start the Skype client: "
      asplaintext="true"></label>
    <skypecall valueprop="skypecall"></skypecall>
  </itr>
</pagebody>
```

Properties

Basic		
valueprop	<div>Name of the adapter parameter that contains the phone number or the Skype ID of the person that should be called. It is also possible to set some parameters.</div> <div>For further information, see the Skype API.</div> <div>Note: The Skype client must be installed if you want to use this control.</div>	Obligatory

79

NJX:BUTTONITEMLIST

▪ Example	529
▪ Adapter Interface	529
▪ Built-in Events	530
▪ Properties	530

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

```
DEFINE DATA PARAMETER
1 DYNBUTTONS (1:*)
2 METHOD (A) DYNAMIC
2 NAME (A) DYNAMIC
2 TITLE (A) DYNAMIC
2 VISIBLE (L)
END-DEFINE
```

Built-in Events

The buttons in the NJX:BUTTONITEMLIST control (NJX:BUTTONITEM controls) behave like **BUTTON** controls.

Properties

Basic			
buttonlistprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
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	

80

NJX:BUTTONITEM

■ Example	532
■ Built-in Events	532
■ Properties	533

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	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	gif jpg jpeg
invisiblemode	<p>This property has three possible values:</p> <p>(1) "invisible": the button is not visible without occupying any space.</p> <p>(2) "disabled": the button is deactivated: it is "grayed" and does not show any roll over effects any more.</p> <p>(3) "cleared": the button is not visible but it still occupies space.</p>	Optional	invisible disabled cleared
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	100 120 140 160 180 200 50% 100%

height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>CSS style definition that is directly passed into the text of this control.</p> <p>With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are:</p> <p>font-weight: bold</p> <p>color: #FF0000</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
buttonstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	Optional	VAR1

	<p>style sheet definition file (.css). If not defined "normal" styles are chosen, if defined (e.g. "VAR1") then other style definitions (xxxVAR1xxx) are chosen.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>		VAR2
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	left center right
valign	<p>Vertical alignment of control in its column.</p> <p>Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes 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.</p>	Optional	top middle bottom
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	1 2 3 4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p>	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
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	

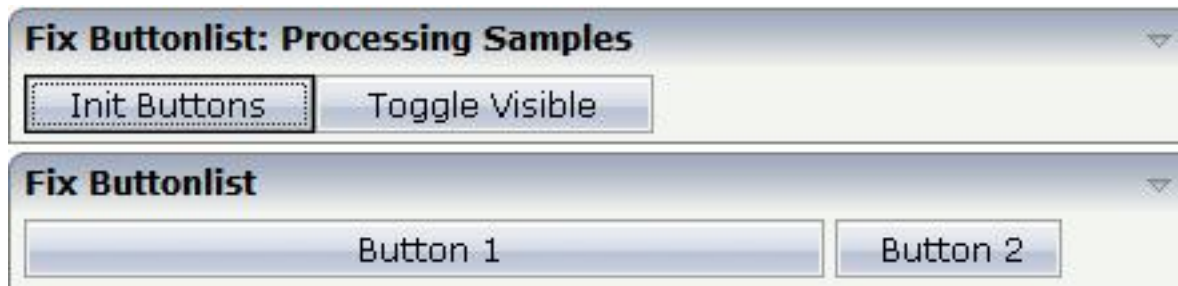
81

NJX:BUTTONITEMLISTFIX

▪ Example	538
▪ Adapter Interface	538
▪ Built-in Events	539
▪ Properties	539

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

```
DEFINE DATA PARAMETER
1 FIXBUTTONS (1:*)
2 METHOD (A) DYNAMIC
2 NAME (A) DYNAMIC
2 TITLE (A) DYNAMIC
2 VISIBLE (L)
END-DEFINE
```


Built-in Events

The buttons in the NJX:BUTTONITEMLISTFIX control (NJX:BUTTONITEMFIX controls) behave like **BUTTON** controls.

Properties

Basic			
buttonlistprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
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	

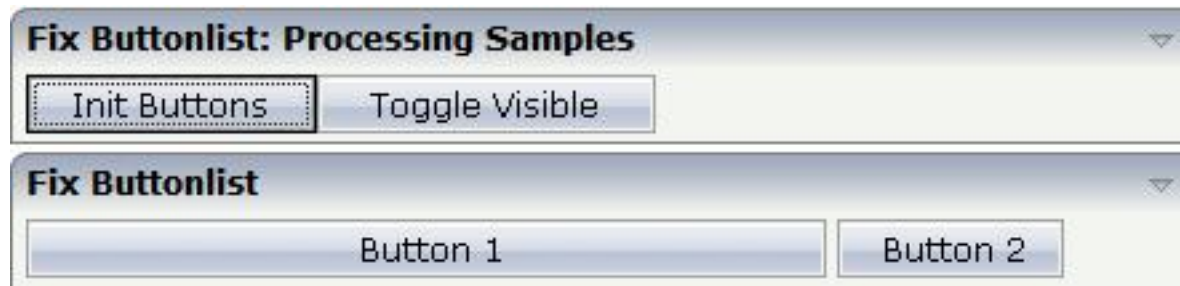
82

NJX:BUTTONITEMFIX

■ Example	542
■ Built-in Events	542
■ Properties	543

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	
method	Name of the event that is sent to the adapter when the user presses the button.	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 button is not visible without occupying any space. (2) "disabled": the button is deactivated: it is "grayed" and does not show any roll over effects any more. (3) "cleared": the button 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:	Optional	100 120 140

	<p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>		160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>CSS style definition that is directly passed into the text of this control.</p> <p>With the style you can individually influence the text of the button. You can specify any style sheet expressions. Examples are:</p> <p>font-weight: bold</p> <p>color: #FF0000</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold

buttonstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
stylevariant	<p>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.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	Optional	VAR1 VAR2
align	<p>Horizontal alignment of control in its column.</p> <p>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.</p> <p>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.</p>	Optional	left center right
valign	<p>Vertical alignment of control in its column.</p> <p>Each control is "packaged" into a column. The column itself is part of a row (e.g. ITR or TR). Sometimtes 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.</p>	Optional	top middle bottom
colspan	Column spanning of control.	Optional	1

	<p>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.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>		<p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>The property only makes sense in table rows that are snychronized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
imagedisabled	<p>URL of image that is displayed if the control is disabled. Use properties VISIBLEPROP and INVISIBLEMODE to disable the control.</p>	Optional	<p>gif</p> <p>jpg</p> <p>jpeg</p>
submitbutton	<p>Set this property to true and the button will work as an 'Submitbutton', that is neccessary if you want to transfer and/or save form values.</p> <p>i.e. password and username or complete search forms</p> <p>Default value is false.</p> <p>You should only use a 'Submitbutton' if the withformtag option of the pagebody tag is set true.</p>	Optional	<p>true</p> <p>false</p>
tabindex	<p>Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.</p>	Optional	<p>-1</p> <p>0</p> <p>1</p> <p>2</p> <p>5</p> <p>10</p>

			32767
Binding			
method	(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 lanaguage 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	

83

NJX:FIELDLIST

▪ Example	551
▪ Adapter Interface	552
▪ Built-in Events	552
▪ Properties	552

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

Complex Field List				
11100102	11100105	11100106	11100107	11100108
Schindler	Schirm	Schmitt	Schmidt	Schneider
Edgar	Christian	Reiner	Helga	Wolfgang

Simple Field List									
Schindl	Schirm	Schmitt	Schmidt	Schneid	Schneid	Bunger	Thiele	Thoma	Treiber

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

```
DEFINE DATA PARAMETER
1 COLUMNS (1:*)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 STATUS (A) DYNAMIC
1 SIMPLE (A/1:*) DYNAMIC
END-DEFINE
```

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			
<code>fieldlistprop</code>	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
<code>fieldcount</code>	Maximum count of fields in the fieldlist. If no <code>fieldcount</code> is defined then a default of 10 is assigned.	Optional	
<code>hdist</code>	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 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.	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 for a generated statusprop variable to which field the statusprop belongs.	Optional	

84

NJX:FIELDITEM

▪ Example	557
▪ Adapter Interface	558
▪ Built-in Events	558
▪ Properties	558

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

Complex Field List				
11100102	11100105	11100106	11100107	11100108
Schindler	Schirm	Schmitt	Schmidt	Schneider
Edgar	Christian	Reiner	Helga	Wolfgang

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

```
DEFINE DATA PARAMETER
1 COLUMNS (1:*)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 STATUS (A) DYNAMIC
END-DEFINE
```

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			
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").	Sometimes obligatory	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%
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 into this FIELD. 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
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

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 synchornized 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 two 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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	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	<p>If the visibility of the control is determined dynamically by an adapter property then there are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>	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)		
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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 representaion directly after changing the value.</p>	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	
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	Optional	

	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.		
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 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 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.	Optional	
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
maxlengthprop	Name of the adapter parameter that provides the maximum number of characters that a user may enter into this FIELD. Consider to use MAXLENGTH to define this number in a static way.	Optional	
Validation			
datatype	<p>By default, the FIELD control is managing its content as string. By explicitly setting a datatype you can define that the control...</p> <p>...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.</p> <p>...will format the data coming from the server or coming from 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).</p>	Optional	date float int long time timestamp color xs:decimal xs:double xs:date

	<p>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.</p> <p>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.</p>		<p>xs:dateTime</p> <p>xs:time</p> <p>-----</p> <p>N n.n</p> <p>P n.n</p> <p>string n</p> <p>xs:byte</p> <p>xs:short</p>
validationrules	<p>Contains information used for Data Validation.</p> <p>Use the Validation Rules Editor to make changes!</p>	Optional	
validation	<p>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.</p>	Optional	<p>[a-zA-Z0-9_-]</p> <p>{1,} \ \ @[a-zA-Z0-9_-]</p> <p>{1,} \ \ . \ \ w{2,} \ \ d{5}</p> <p>[0-9](-/+)+</p>
validationprop	<p>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.</p>	Optional	
validationuserhint	<p>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.</p>	Optional	
validationuserhintprop	<p>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.</p>	Optional	
digits	<p>Number that specifies how many digits are to be displayed (ie digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.</p>	Optional	<p>1</p> <p>2</p>

			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	Number that specifies how many decimal digits are to be displayed. 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	
spinrangemax	An integer value which defines the upper bound of the value range.	Optional	
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. See at chapter 'Popup Dialog Management' for more details. 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 openIdValueComboOrPopup
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	Optional	

	in which a FIELD offers e.g. value help or not, depending on business logic inside the adapter.		
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.	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 lanaguage 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	Comma 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 comma 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.	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 for 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	

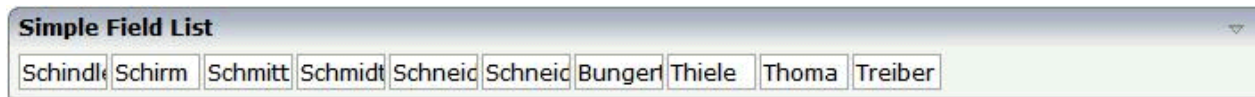
85

NJX:FIELDVALUE

▪ Example	571
▪ Adapter Interface	571
▪ Built-in Events	571
▪ Properties	572

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

```
DEFINE DATA PARAMETER
1 SIMPLE (A/1:*) DYNAMIC
END-DEFINE
```

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Sometimes obligatory	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p> <p>50%</p> <p>100%</p>
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	<p>5</p> <p>10</p> <p>15</p> <p>20</p> <p>int-value</p>
maxlength	Maximum number of characters that a user may enter into this FIELD. 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	<p>5</p> <p>10</p> <p>15</p> <p>20</p>

			int-value
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
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	Optional	1 2 3

	<p>"1" - but you may want to define the control to span over more than one columns.</p> <p>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.</p>		<p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>4</p> <p>5</p> <p>50</p> <p>int-value</p>
fieldstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
noborder	<p>Boolean value defining if the control has a border. Default is "false".</p>	Optional	<p>true</p> <p>false</p>
transparentbackground	<p>Boolean value defining if the control is rendered with a transparent background. Default is "false".</p>	Optional	<p>true</p> <p>false</p>
invisiblemode	<p>If the visibility of the control is determined dynamically by an adapter property then there</p>	Optional	<p>invisible</p>

	<p>are two rendering modes if the visibility is "false":</p> <p>(1) "invisible": the control is not visible.</p> <p>(2) "disabled": the control is deactivated: it is "grayed" and does not show any roll over effects any more.</p>		cleared
tabindex	<p>Index that defines the tab order of the control. Controls are selected in increasing index order and in source order to resolve duplicates.</p>	Optional	<p>-1</p> <p>0</p> <p>1</p> <p>2</p> <p>5</p> <p>10</p> <p>32767</p>
Binding			
flush	<p>Flushing behaviour of the input control.</p> <p>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.</p> <p>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.</p> <p>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.</p>	Optional	<p>screen</p> <p>server</p>
flushmethod	<p>When the data synchronization of the control is set to FLUSH="server" then you can specify an</p>	Optional	

	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.		
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	
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 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.	Optional	
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
maxlengthprop	Name of the adapter parameter that provides the maximum number of characters that a user may enter into this FIELD. 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 from the user input: if the field has datatype "date" and the user inputs "010304" then the input will be translated into	Optional	date float int long time timestamp color xs:decimal

	<p>"01.03.2004" (or other representation, dependent on date format settings).</p> <p>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.</p> <p>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.</p>		xs:double xs:date xs:dateTime xs:time ----- N n.n P n.n string n xs:byte xs:short
validationrules	<p>Contains information used for Data Validation.</p> <p>Use the Validation Rules Editor to make changes!</p>	Optional	
validation	<p>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.</p>	Optional	[a-zA-Z0-9_.-] {1,}\ \ @[a-zA-Z0-9_.-] {1,}\ \ .\ \ w{2,}\ \ d{5} [0-9](-/+)+
validationuserhint	<p>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.</p>	Optional	
validationuserhintprop	<p>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.</p>	Optional	
digits	<p>Number that specifies how many digits are to be displayed (ie digits before the comma). If using this feature then the DATATYPE property must be set to 'float'. See also DECIMALDIGITS.</p>	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	Number that specifies how many decimal digits are to be displayed. 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	
spinrangemax	An integer value which defines the upper bound of the value range.	Optional	
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. See at chapter 'Popup Dialog Management' for more details. 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 openIdValueComboOrPopup
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 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 lanaguage 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	Comma 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 comma 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	

86

NJX:NJXVARIABLE

▪ Example	582
▪ Properties	582

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	

87

NJX:EVENTDATA

▪ Example	585
▪ Adapter Interface	586

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

```
DEFINE DATA PARAMETER
1 LINES (1:*)
2 FIRST (A) DYNAMIC
2 ID (A) DYNAMIC
2 LAST (A) DYNAMIC
2 SELECTED (L)
1 XCIEVENTDATA
2 XCIINDEX (I4)
END-DEFINE
```

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`.

■ Properties	588
--------------------	-----

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			
translationreference	<p>This is the "translation reference" that is passed to the multi language management.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>If you on the other hand define one translation reference for multiple pages then you can share text-ids throughout the various pages.</p> <p>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 if it is clear what they stand for. E.g. you may name buttons in the following way: "btn_save" and "btn_saveas".</p>	Sometimes obligatory	
stylesheetfile	<p>URL of a style sheet file used for control rendering.</p> <p>Typically the style sheet file used for control rendering is set dynamically e.g. the style depends on the user who is</p>	Optional	css

	<p>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.</p> <p>The style sheet file must be defined as URL, relative to the generated page. A valid value may be <code>"../softwareag/styles/CIS_DEFAULT.css"</code>.</p> <p>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:</p> <p>(1) The adapter object provides for a <code>"String getStyle()"</code> method that return the URL. You can override the default method and pass back your own URL.</p> <p>(2) When using the default implementation derived from <code>com.softwareag.cis.server.Model</code> then the <code>getStyle()</code> method accesses the CIS session context. You can set the session's style by calling <code>"findCISessionContext()"</code> in your adapter and calling <code>"setStyle()"</code> in the session context's object.</p>		
addstylesheetfile	<p>URL of an additional style sheet file.</p> <p>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:</p> <p>(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).</p> <p>(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.</p> <p>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".</p>	Optional	css
imagestopreload	<p>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.</p>	Optional	

	<p>The URL of the images must be relative to your generated HTML page.</p> <p>Example: if your page has a tree with certain node images then you may define: "images/nodeopened.gif" images/nodeclosed.gif; images/nodeendnode.gif".</p>		
darkbackground	<p>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.</p> <p>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.</p>	Optional	true false
helpid	<p>This is the id that is passed into the help management for the page.</p> <p>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.</p>	Optional	
visiblevalueifundefined	<p>Several CIS controls support a VISIBLEPROP property. The VISIBLEPROP contains the binding to an adapter property that decides at runtime if a control is visible or not.</p> <p>This property defines how these controls behave if there is no implementation available for the property.</p> <p>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.</p>	Optional	true false
contextmenumethod	<p>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. texgrid, tree,...) handled the click so far.</p>	Optional	
immediatedisplay	<p>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.</p>	Optional	true false
addjavascriptlibs	<p>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.</p>	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
usexmlhttprequest	By default CIS framework is using hidden frame communication (asynchronous server communication). Use this attribute in order to use "XMLHttpRequests". Typical usage is with timer pages (to avoid seeing ongoing communication to server on browser's statusbar).	Optional	
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	
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	
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
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	Optional	1 2 3

	recursion depth of these structures can be limited using this property.		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 CIS page can be opened as a popup dialog. This properties define the pixel width preferred for the page. - See the property "popupheight" for more information.	Optional	100px 200px 300px 400px
popupheight	Each CIS page can be opened as a popup dialog. This property defines the pixel height preferred for the page. A popup is typically opened by calling the "openPopup"-method in your adapter code. If no further definition is done then the popup will open in the height that is defined by this value. You can also dynamically manipulate the size and position of the popup by using the Model-method "setPopupFeatures" - please read corresponding documentation inside the Java API documentation.	Optional	100px 200px 300px 400px
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.	Optional	dialogLeft: 200px dialogTop: 100px

	<p>center:yes no</p> <p>edge:sunken raised</p> <p>resizable:yes no</p> <p>scroll:yes no</p> <p>status:yes no (to display or hide a status bar)</p> <p>An example string looks as follows: "dialogLeft:100px"</p> <p>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.</p> <p>Please also pay attention to the methods "setPopupTitle()" and "setPopupPageFeatures()" in the com.casabac.server.Model class. By using these method you can define popup parameters in a dynamic way inside your adapter implementation.</p>		<p>edge: sunken</p> <p>resizable: yes</p> <p>status: no</p>
Occupied			
occupiedimage	<p>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.</p> <p>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.</p>	Optional	
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	Comma 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 comma	Optional	

	<p>Example:</p> <p>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.</p> <p>Use the popup help within the Layout Painter to input hot keys.</p>		
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	

89

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**
- **MGDGRID - Managing the Grid**
- **GRIDCOLHEADER - Flexible Column Headers**

90

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.

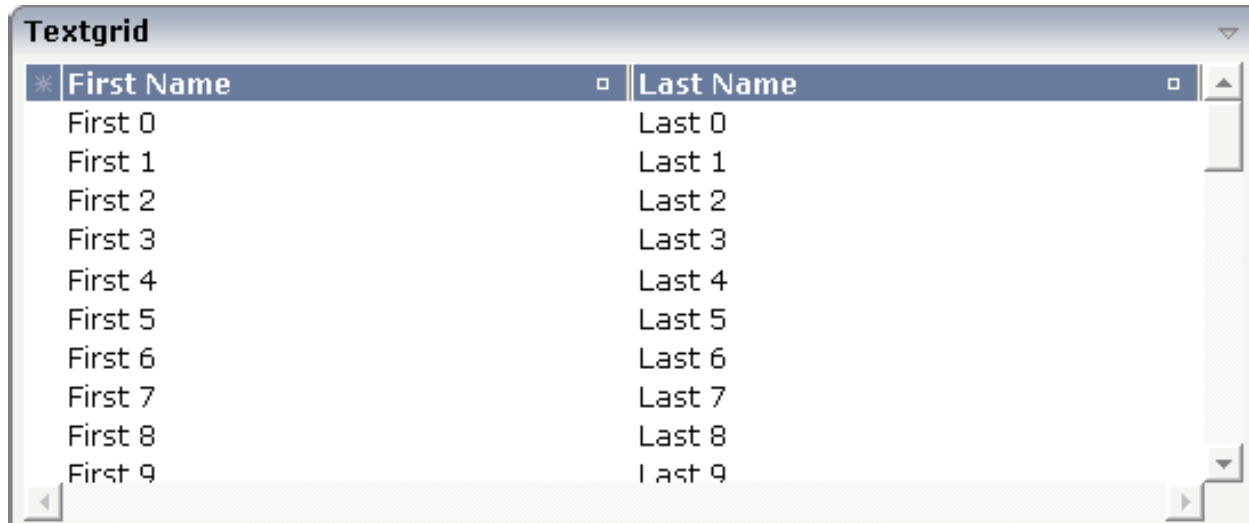
91

TEXTGRID2

■ A Simple Example	600
■ Adapter Interface	601
■ Selecting Rows in a TEXTGRID2	601
■ TEXTGRID2 Properties	602
■ COLUMN Properties	608
■ Dynamic Setting of Text Styles in TEXTGRID2	612

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

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
END-DEFINE
```

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Obligatory	100
			120
			140
			160
			180
			200
			50%
			100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>Name of an adapter parameter that is used to mark if an individual row of the text grid should receive the focus.</p> <p>If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.</p>	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	<p>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.</p> <p>Default is "false".</p>	Optional	true false noselection
enabledefaultcontextmenu	<p>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.</p> <p>Default is "false".</p>	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	<p>Definition of the horizontal scrollbar's appearance.</p> <p>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").</p> <p>Default is "auto".</p>	Sometimes obligatory	auto scroll hidden
withtitlerow	<p>If defined as "false" then no top title row is shown.</p> <p>"True" is default.</p>	Optional	true false

colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
personalizable	<p>If defined to "false" then no re-arranging of columns is offered to the user.</p> <p>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.</p>	Optional	true false
stylevariant	<p>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.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	Optional	VAR1 VAR2
backgroundstyle	CSS style definition that is directly passed into this control.	Optional	

	<p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		
vscroll	<p>Definition of the vertical scrollbar's appearance.</p> <p>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").</p> <p>Default is "auto".</p>	Optional	auto scroll hidden
withrollover	<p>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).</p>	Optional	true false
fixedcolumnsizes	<p>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.</p>	Optional	true false
requiredheight	<p>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%).</p> <p>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.</p>	Optional	1 2 3 int-value
disablecolumnresizing	<p>Flag that indicates if the user can change the width of the grid columns. Default is false.</p>	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			
draginfo	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 (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.	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 for a generated statusprop variable to which field the statusprop belongs.	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.

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).	Optional	date float int long

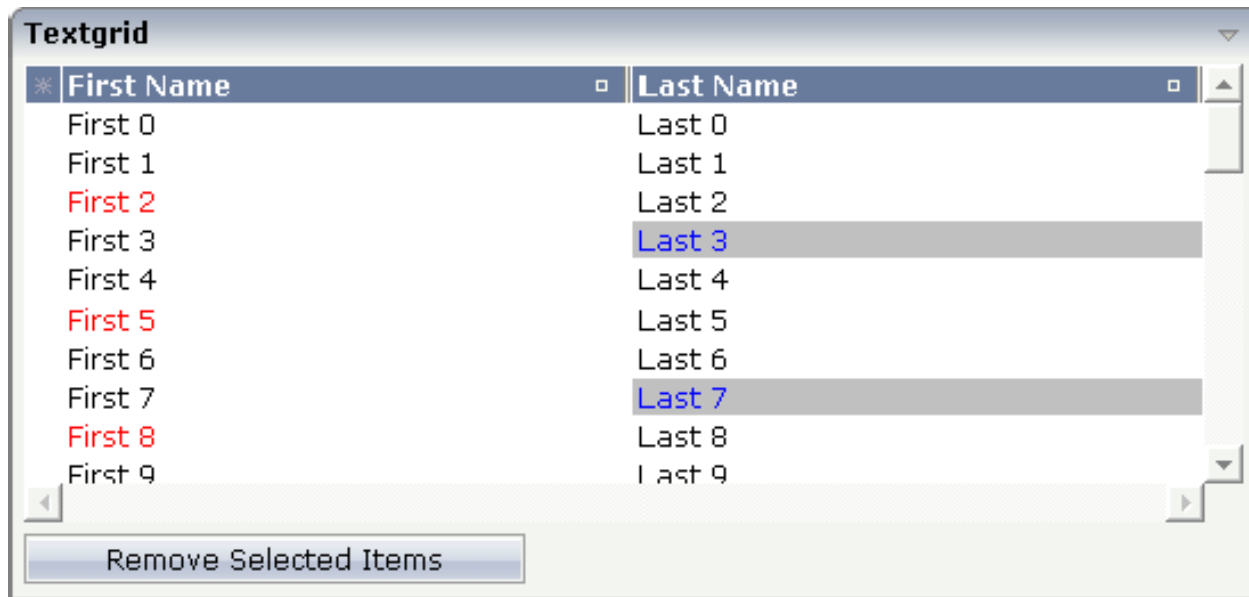
	Please note: the datatype "float" is named a bit misleading - it represents any decimal format number. The server side representation may be a float value, but also can be a double or a BigDecimal property.		time timestamp color xs:decimal xs:double xs:date xs:dateTime xs:time ----- N n.n P n.n string n xs:byte xs:short
align	Horizontal alignment of the control's content.	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". MOZILLA: this property is not available in Mozilla!	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 (andnbsp\"). 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	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	
Binding			
property	(already explained above)		
textstyleprop	<p>Name of the adapter parameter that provides a style-string that is used for rendering the column's content.</p> <p>As consequence you can individually assign a CSS-style to each cell of your text grid.</p>	Optional	
textclassprop	<p>Name of the adapter parameter that provides a style class to be used for rendering the content.</p> <p>You can set up a limited number of style classes inside your style sheet definition - and dynamically reference them per grid cell.</p>	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	
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.	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 lanaguage 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 lanaguage management - representing the tooltip text for the sort indicator.	Optional	
celltitleprop	(already explained above)		
Natural			
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: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: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.	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: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 for a generated statusprop variable to which field the statusprop belongs.	Optional	
njx:natcv	Name of a Natural control variable that shall be assigned to the control. 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 control.	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>
```

■ Performance Considerations	614
■ Example	614
■ Adapter Interface	616
■ Using Server-Side Scrolling	616
■ Using Server-Side Sorting	617
■ TEXTGRIDSSS2 Properties	617

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". What this means is described in this chapter.

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.

Textgridsss2	
* First Name	Last Name
First 0	Last 0
▶ First 1	Last 1
▶ First 2	Last 2
First 3	Last 3
First 4	Last 4
First 5	Last 5
First 6	Last 6
▶ First 7	Last 7
First 8	Last 8
First 9	Last 9
First 10	Last 10
▶ First 11	Last 11
First 12	Last 12
First 13	Last 13
First 14	Last 14
▶ First 15	Last 15
First 16	Last 16
First 17	Last 17
First 18	Last 18
First 19	Last 19

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>
```

Adapter Interface

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
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)
END-DEFINE
```

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.

TEXTGRIDSSS2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the grid in the adapter.	Obligatory	
rowcount	<p>Number of rows that is rendered inside the control.</p> <p>There are two ways of using this property - dependent on whether you in addition define the HEIGHT property:</p> <p>If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that is defined as ROWCOUNT value.</p> <p>If a HEIGHT value is defined in addition (e.g. as percentage value "100%") then the number of rows depends on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that is picked from the server. You should define this value in a way 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.</p>	Obligatory	
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p>	Obligatory	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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
onloadbehaviour	<p>Loading behaviour of the items into the client.</p> <p>"block" (=default) means that the client always requests the currently visible items from the server (=Server Side Scrolling).</p> <p>"collection" means that the client requests all items at the beginning from the server. The client itself implements the scrolling in the JavaScript/SWT.</p>	Optional	block collection
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.	Optional	

	If the user selects a text grid row, the value "true" is passed into the corresponding array element of the adapter parameter.		
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 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	
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 "auto".	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 "auto".	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. "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	Optional	1 2 3

	<p>occupies. By default it is "1" - but you may want to define the control to span over more than one columns.</p> <p>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.</p>		4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
personalizable	<p>If defined to "false" then no re-arranging of columns is offered to the user.</p> <p>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.</p>	Optional	true false
stylevariant	<p>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.</p> <p>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 offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	Optional	VAR1 VAR2
backgroundstyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p>	Optional	

	<p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>		
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	<p>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%).</p> <p>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.</p>	Optional	1 2 3 int-value
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
showemptylines	If set to false, no empty line will be rendered. By default empty lines are shown.	Optional	true false
withsliderfreeze	Setting this to "true" prevents unwished slider jumps while scrolling up/down in a grid with a huge number of lines (for example 20000).	Optional	true false
Drag And Drop			
draginfo	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 (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.	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 for a generated statusprop variable to which field the statusprop belongs.	Optional	
Deprecated			

directselectmethod	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	
directselectevent	Use ONCLICKMETHOD and ONDBLCLICKMETHOD instead.	Optional	ondblclick onclick

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.

93

ROWTABLEAREA2 - The Flexible Control Grid

■ Example	626
■ Adapter Interface	628
■ Built-in Events	628
■ Making Grids Look like Grids	629
■ ROWTABLEAREA2 Properties	630
■ STR Properties	636

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 example library SYSEXNJX.

Example

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.

	First Name	Last Name
<input type="checkbox"/>	First 1	Last 1
<input checked="" type="checkbox"/>	First 2	Last 2
<input checked="" type="checkbox"/>	First 3	Last 3
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		
<input type="checkbox"/>		

Add new Line Remove selected Lines

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>
  </rowtablearea2>
</rowarea>
```



```

        </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

In the parameter data area of the adapter, the grid data is represented by the following data structure:

```
DEFINE DATA PARAMETER
1 LINES (1:*)
2 FIRSTNAME (U) DYNAMIC
2 LASTNAME (U) DYNAMIC
2 SELECTED (L)
END-DEFINE
```

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".

```
DEFINE DATA PARAMETER
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)
END-DEFINE
```

Built-in Events

value-of-griddataprop.onCtrlSelect
value-of-griddataprop.onSelect
value-of-griddataprop.onShiftSelect
value-of-griddataprop.onSort
value-of-griddataprop.onTopindexChanged

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:

	Article	Price
<input type="checkbox"/>	Article 1	0.99
<input type="checkbox"/>	Article 2	1.98
<input type="checkbox"/>	Article 3	2.97
<input type="checkbox"/>	Article 4	3.96
<input type="checkbox"/>	Article 5	4.96
<input type="checkbox"/>	Article 6	5.94
<input type="checkbox"/>	Article 7	6.93
<input type="checkbox"/>	Article 8	7.92
<input type="checkbox"/>	Article 9	8.92
<input type="checkbox"/>	Article 10	9.91

This is collection of fields:

	Article	Price
<input type="checkbox"/>	Article 1	0.99
<input type="checkbox"/>	Article 2	1.98
<input type="checkbox"/>	Article 3	2.97
<input type="checkbox"/>	Article 4	3.96
<input type="checkbox"/>	Article 5	4.96
<input type="checkbox"/>	Article 6	5.94
<input type="checkbox"/>	Article 7	6.93
<input type="checkbox"/>	Article 8	7.92
<input type="checkbox"/>	Article 9	8.92
<input type="checkbox"/>	Article 10	9.91

ROWTABLEAREA2 Properties

Basic			
griddataprop	Name of the adapter parameter that represents the control in the adapter.	Obligatory	
rowcount	<p>Number of rows that is renders inside the control.</p> <p>There are two ways of using this property - dependent on whether you in addition define the HEIGHT property:</p> <p>If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that is defined as ROWCOUNT value.</p> <p>If a HEIGHT value is defined an addition (e.g. as percentage value "100%") then the number of rows depends on the actual height of the control. The ROWCOUNT value in this case indicates</p>	Optional	

	the maximum number of rows that is picked from the server. You should define this value in a way 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	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	Optional	100 150 200 250 300 250 400 50% 100%
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p>	Sometimes obligatory	100 120 140 160 180 200 50% 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.		
firstrowcolwidths	<p>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.</p> <p>Default is "false", i.e. the grid is sized according to its "whole content".</p> <p>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.</p>	Sometimes obligatory	true false
onloadbehaviour	<p>Loading behaviour of the items into the client.</p> <p>"block" (=default) means that the client always requests the currently visible items from the server (=Server Side Scrolling).</p> <p>"collection" means that the client requests all items at the beginning from the server. The client itself implements the scrolling in the JavaScript/SWT.</p>	Optional	block collection
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
withborder	<p>If set to "false" then no thin border is drawn around the controls that are contained in the grid.</p> <p>Default is "true".</p>	Optional	true false

hscroll	<p>Definition of the horizontal scrollbar's appearance.</p> <p>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").</p> <p>Default is "auto".</p>	Optional	<p>auto</p> <p>scroll</p> <p>hidden</p>
vscroll	<p>Definition of the vertical scrollbar's appearance.</p> <p>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").</p> <p>Default is "auto".</p>	Optional	<p>auto</p> <p>scroll</p> <p>hidden</p>
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	<p>true</p> <p>false</p>
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	<p>true</p> <p>false</p>
touchpadinput	<p>If set to "true" then touch screen icons for scrolling are displayed in addition.</p> <p>Default is "false".</p>	Optional	<p>true</p> <p>false</p>
requiredheight	<p>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%).</p> <p>Please note: You must not use FIXLAYOUT at the surrounding row container (ITR and ROWAREA). Otherwise: if the available space is less</p>	Optional	<p>1</p> <p>2</p> <p>3</p> <p>int-value</p>

	than the required height the end of the control is just cut off.		
tablestyle	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
darkbackground	<p>Normally the background is in light colour but the CIS style sheets also have a dark(er) grey colour to be used.</p> <p>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.</p>	Optional	<p>true</p> <p>false</p>
invisiblemodeincompletelastrow	If set to "invisible" an incomplete last row is not shown.	Optional	<p>invisible</p> <p>visible</p>
withsliderfreeze	Setting this to "true" prevents unwished slider jumps while scrolling up/down in a grid with a huge number of lines (for example 20000).	Optional	<p>true</p> <p>false</p>
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	
Hot Keys			
hotkeys	<p>Comma 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 comma</p> <p>Example:</p> <p>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.</p> <p>Use the popup help within the Layout Painter to input hot keys.</p>	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	Optional	

	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.		
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 for 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 a line.	Optional	
ondblclickmethod	Name of the event that is sent to the adapter when the user double clicks a line.	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 be passed.	Optional	
backgroundcolorprop	Name of the adapter parameter that dynamically provides the background color for this control.	Optional	

94

MGDGRID - Managing the Grid

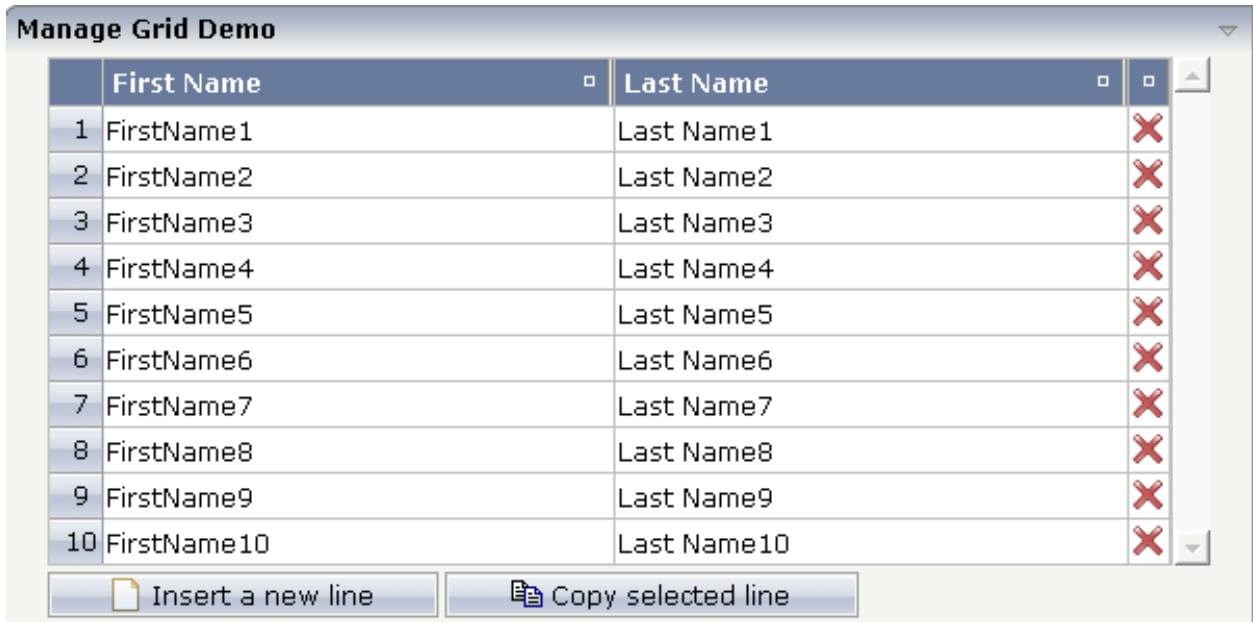
■ Example	641
■ Adapter Interface	642
■ Built-in Events	643
■ MGDGRID Properties	643
■ ROWINSERT Properties	647
■ ROWCOPY Properties	648
■ ROWDELETE Properties	649

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 example library SYSEXNJX. 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:

```
DEFINE DATA PARAMETER
1 MGLINES (1:*)
2 FNAME (U) DYNAMIC
2 LNAME (U) DYNAMIC
2 SELECTED (L)
END-DEFINE
```

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".

```
DEFINE DATA PARAMETER
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)
END-DEFINE
```

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	<p>Number of rows that is renderes inside the control.</p> <p>There are two ways of using this property - dependent on whether you in addition define the HEIGHT property:</p> <p>If you do NOT define the HEIGHT property then the control is rendered with exactly the number of rows that is defined as ROWCOUNT value.</p> <p>If a HEIGHT value is defined an addition (e.g. as percentage value "100%") then the number of rows depends on the actual height of the control. The ROWCOUNT value in this case indicates the maximum number of rows that is picked from the server. You should define this value in a way 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.</p>	Optional	
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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</p>	Optional	<p>100</p> <p>150</p> <p>200</p> <p>250</p>

	<p>(containing) other controls then the height of the control will follow the height of its content.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>		<p>300</p> <p>250</p> <p>400</p> <p>50%</p> <p>100%</p>
width	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Sometimes obligatory	<p>100</p> <p>120</p> <p>140</p> <p>160</p> <p>180</p> <p>200</p> <p>50%</p> <p>100%</p>
firstrowcolwidths	<p>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.</p> <p>Default is "false", i.e. the grid is sized according to its "whole content".</p> <p>Please note: when using the GRIDCOLHEADER control within the header-TR-row this property</p>	Sometimes obligatory	<p>true</p> <p>false</p>

	must be set to "true" - otherwise column resizing (by drag and drop) does not work correctly.		
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 "auto".	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 "auto".	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>
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 very first 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	<p>Comma 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 comma</p> <p>Example:</p> <p>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.</p> <p>Use the popup help within the Layout Painter to input hot keys.</p>	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.	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 for a generated statusprop variable to which field the statusprop belongs.	Optional	

ROWINSERT Properties

Basic			
image	<p>URL that points to the image that is shown as icon.</p> <p>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.</p>	Obligatory	

	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.		
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 lanaguage 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 lanaguage management - representing the tooltip text that is used for the control.	Optional	

ROWDELETE Properties

Basic			
image	<p>URL that points to the image that is shown as icon.</p> <p>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.</p> <p>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.</p>	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	<p>Text that is shown as tooltip for the control.</p> <p>Either specify the text "hard" by using this TITLE property - or use the TITLETEXTID in order to define a language dependent literal.</p>	Optional	
titletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text that is used for the control.	Optional	

95

GRIDCOLHEADER - Flexible Column Headers

▪ Flexible Column Sizing	652
▪ Flexible Column Sorting	655
▪ GRIDCOLHEADER Properties	656
▪ 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 were 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:

Grid Col Header Example

<input type="checkbox"/> First Name <input type="checkbox"/>	<input type="checkbox"/> Last Name <input type="checkbox"/>	
<input type="checkbox"/> Last Name 0	Last Name 0	
<input type="checkbox"/> Last Name 1	Last Name 1	
<input type="checkbox"/> Last Name 2	Last Name 2	
<input type="checkbox"/> Last Name 3	Last Name 3	
<input type="checkbox"/> Last Name 4	Last Name 4	
<input type="checkbox"/> Last Name 5	Last Name 5	
<input type="checkbox"/> Last Name 6	Last Name 6	
<input type="checkbox"/> Last Name 7	Last Name 7	
<input type="checkbox"/> Last Name 8	Last Name 8	
<input type="checkbox"/> Last Name 9	Last Name 9	

Grid Col Header Example

<input type="checkbox"/> First Name <input type="checkbox"/>	<input type="checkbox"/> Last Name <input type="checkbox"/>
<input type="checkbox"/> Last Name 0	Last Name 0
<input type="checkbox"/> Last Name 1	Last Name 1
<input type="checkbox"/> Last Name 2	Last Name 2
<input type="checkbox"/> Last Name 3	Last Name 3
<input type="checkbox"/> Last Name 4	Last Name 4
<input type="checkbox"/> Last Name 5	Last Name 5
<input type="checkbox"/> Last Name 6	Last Name 6
<input type="checkbox"/> Last Name 7	Last Name 7
<input type="checkbox"/> Last Name 8	Last Name 8
<input type="checkbox"/> Last Name 9	Last Name 9

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.

□	First Name	□	Last Name	□
<input type="checkbox"/>	Last Name 0		Last Name 0	
<input type="checkbox"/>	Last Name 1		Last Name 1	
<input type="checkbox"/>	Last Name 2		Last Name 2	

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 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%
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.	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 lanaguage 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
image	<p>URL of image that is displayed inside the control. Any image type (.gif, .jpg, ...) that your browser does understand is valid.</p> <p>Use the following options to specify the URL:</p> <p>(A) Define the URL relative to your page. Your page is generated directly into your project's folder. Specifiying "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.</p> <p>(B) Define a complete URL, like "http://www.softwareag.com/images/logo.gif".</p>	Optional	
stylevariant	<p>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.</p> <p>Purpose: you can set up style variants in the style sheet defintion and use them multiple times by addressing them via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!</p>	Optional	VAR1 VAR2 VAR3 VAR4
sorttitle	<p>Text that is shown as tooltip for the sort indicator.</p> <p>Either input text by using this SORTTITLE property - or use the SORTTITLETEXTID in order to define a language dependent literal.</p>	Optional	
sorttitletextid	Text ID that is passed to the multi lanaguage management - representing the tooltip text for the sort indicator.	Optional	
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	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>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.</p>	Optional	1 2 3 4 5 50 int-value
colspan	<p>Column spanning of control.</p> <p>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.</p> <p>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.</p>	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	
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:

<input type="checkbox"/>	First Name	<input type="checkbox"/>	Last Name	<input type="checkbox"/>
	Last Name 0		Last Name 0	
	Last Name 1		Last Name 1	
	Last Name 2		Last Name 2	

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>
```

```
</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	\$en/popupwizard/njx_selector_attr_valueprop\$	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%
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	<p>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").</p> <p>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.</p>	Optional	true false
image	<p>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.</p> <p>If specifying WITHLINENUM to be "true" then the line number of selected lines is output in bold font.</p>	Optional	
imageprop	\$en/popupwizard/njx_selector_attr_imageprop\$	Optional	
alwaysshowicon	<p>Flag that indicates if the selector shows its image - independent from whether the corresponding line is selected or not. With ALWAYSSHOWICON you can show icons on unselected lines, too. For that specify WITHLINENUM to be "false" and use IMAGEPROP.</p> <p>Default is "false".</p>	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	

96

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**

97

Basics





■ Types of Trees	666
■ When to Use Which Type	667

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.).

Tree			
Tree Node		Toggle Count	Select Count
<input type="checkbox"/>  Top		0	0
<input type="checkbox"/> Sub 1		0	0
<input type="checkbox"/>  Sub 2		1	0
<input type="checkbox"/>  Sub 2-1		1	0
<input type="checkbox"/> Sub 2-1-1		0	0
<input type="checkbox"/> Sub 2-1-2		0	0
<input checked="" type="checkbox"/>  Sub 2-2		0	0
<input type="checkbox"/> Sub 3		0	0

Of course, you do not have to combine it with other controls. You can also use it “stand-alone” inside a ROWTABLEAREA2 grid:

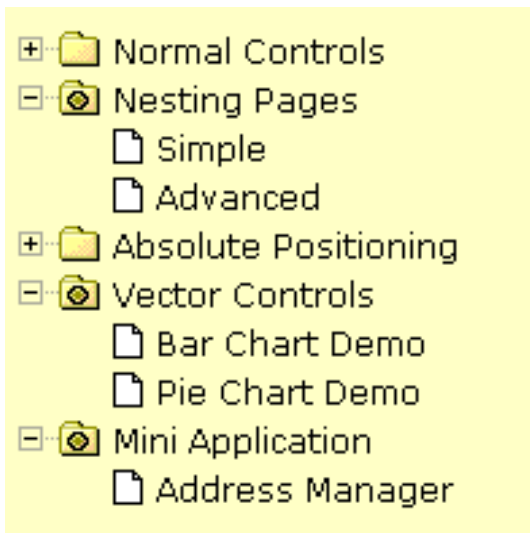
<input type="checkbox"/>  page
<input checked="" type="checkbox"/>  titlebar (Template)
<input type="checkbox"/>  header
<input type="checkbox"/>  button (Save)
<input checked="" type="checkbox"/>  pagebody
<input type="checkbox"/>  statusbar

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.

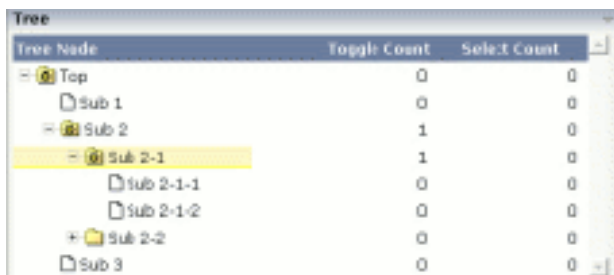
98

TREENODE3 in Control Grid (ROWTABLEAREA2)

■ Example	670
■ Adapter Interface	671
■ Built-in Events	671
■ Properties	671

Example

The following image shows an example for a tree management:



Tree Node	Toggle Count	Select Count
Top	0	0
Sub 1	0	0
Sub 2	1	0
Sub 2-1	1	0
Sub 2-1-1	0	0
Sub 2-1-2	0	0
Sub 2-2	0	0
Sub 3	0	0

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:

```
DEFINE DATA PARAMETER
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
END-DEFINE
```

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	Optional	1 2 3 int-value

	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.		
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: 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. In this case the select() method is called in the corresponding node object on server side.	Optional	ondblclick onclick
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 offerst 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 offerst 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	Optional	1 2

	<p>it is "1" - but you may want to define the control to span over more than one columns.</p> <p>The property only makes sense in table rows that are synchornized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>		3 4 5 50 int-value
rowspan	<p>Row spanning of control.</p> <p>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 two span over more than one columns.</p> <p>The property only makes sense in table rows that are synchornized within one container (i.e. TR, STR table rows). It does not make sense in ITR rows, because these rows are explicitly not synched.</p>	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	<p>Name of an adapter parameter that provides for a image for the tree node.</p> <p>Each node may provide for its own image, e.g. dependent on the type of node.</p>	Optional	

	If the adapter property passes back an empty string, then the image is taken from the static definitions that you may parallelly do by 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

99 CLIENTTREE

▪ Example	678
▪ Adapter Interface	679
▪ Built-in Events	679
▪ Properties	679

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:

```
DEFINE DATA PARAMETER
1 TREE (1:*)
2 LEVEL (I4)
2 OPENED (I4)
2 SELECTED (L)
2 TEXT (U) DYNAMIC
END-DEFINE
```

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.	Optional	100
	There are three possibilities to define the height:		150
	(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.		200
			250
	(B) Pixel sizing: just input a number value (e.g. "20").		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		400
			50%
			100%

	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			
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	Optional	VAR1 VAR2

	via the "stylevariant" property. CIS currently offerst two variants "VAR1" and "VAR2" but does not predefine any semantics behind - this is up to you!		
hscroll	<p>Definition of the horizontal scrollbar's appearance.</p> <p>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").</p> <p>Default is "auto".</p>	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	Optional	true false

	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.		
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 (background of the client tree). You can as consequence e.g. define the background-colour of the tree dependent on your server side logic.	Optional	
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	
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

100

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**

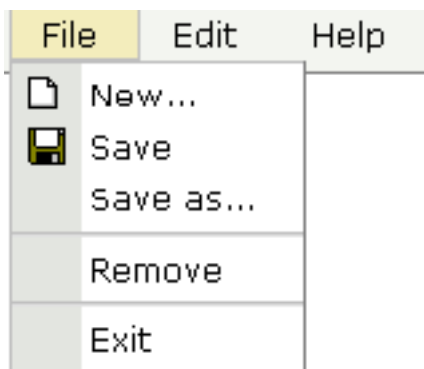
101

Types of Menus

The following menu controls are available:

- **MENU**

This is the typical drop-down menu:



- **DLMENU**

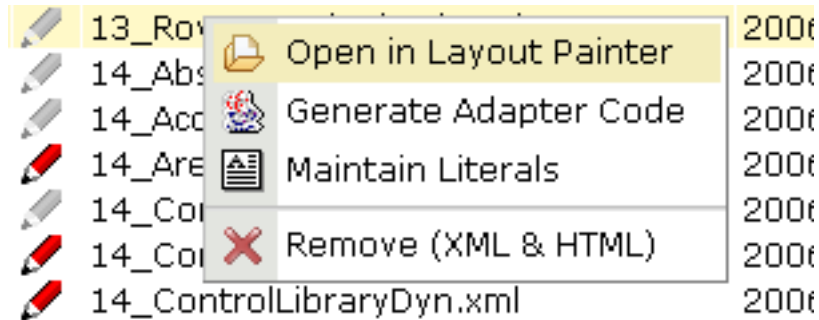
This is a double-line menu representing a two-level hierarchy. It can be found quite often in web applications.



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.

102

MENU

▪ Example	688
▪ Adapter Interface	689
▪ Built-in Events	689
▪ Properties	690

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="Menu_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

```
DEFINE DATA PARAMETER
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
END-DEFINE
```

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	<p>Width of the control.</p> <p>There are three possibilities to define the width:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "100").</p> <p>(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.</p>	Optional	100 120 140 160 180 200 50% 100%
height	<p>Height of the control.</p> <p>There are three possibilities to define the height:</p> <p>(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.</p> <p>(B) Pixel sizing: just input a number value (e.g. "20").</p> <p>(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.</p>	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	<p>CSS style definition that is directly passed into this control.</p> <p>With the style you can individually influence the rendering of the control. You can specify any style sheet expressions. Examples are:</p> <p>border: 1px solid #FF0000</p> <p>background-color: #808080</p> <p>You can combine expressions by appending and separating them with a semicolon.</p> <p>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.</p>	Optional	
menustyleprop	Name of the adapter parameter that dynamically provides explicit style information for the control.	Optional	

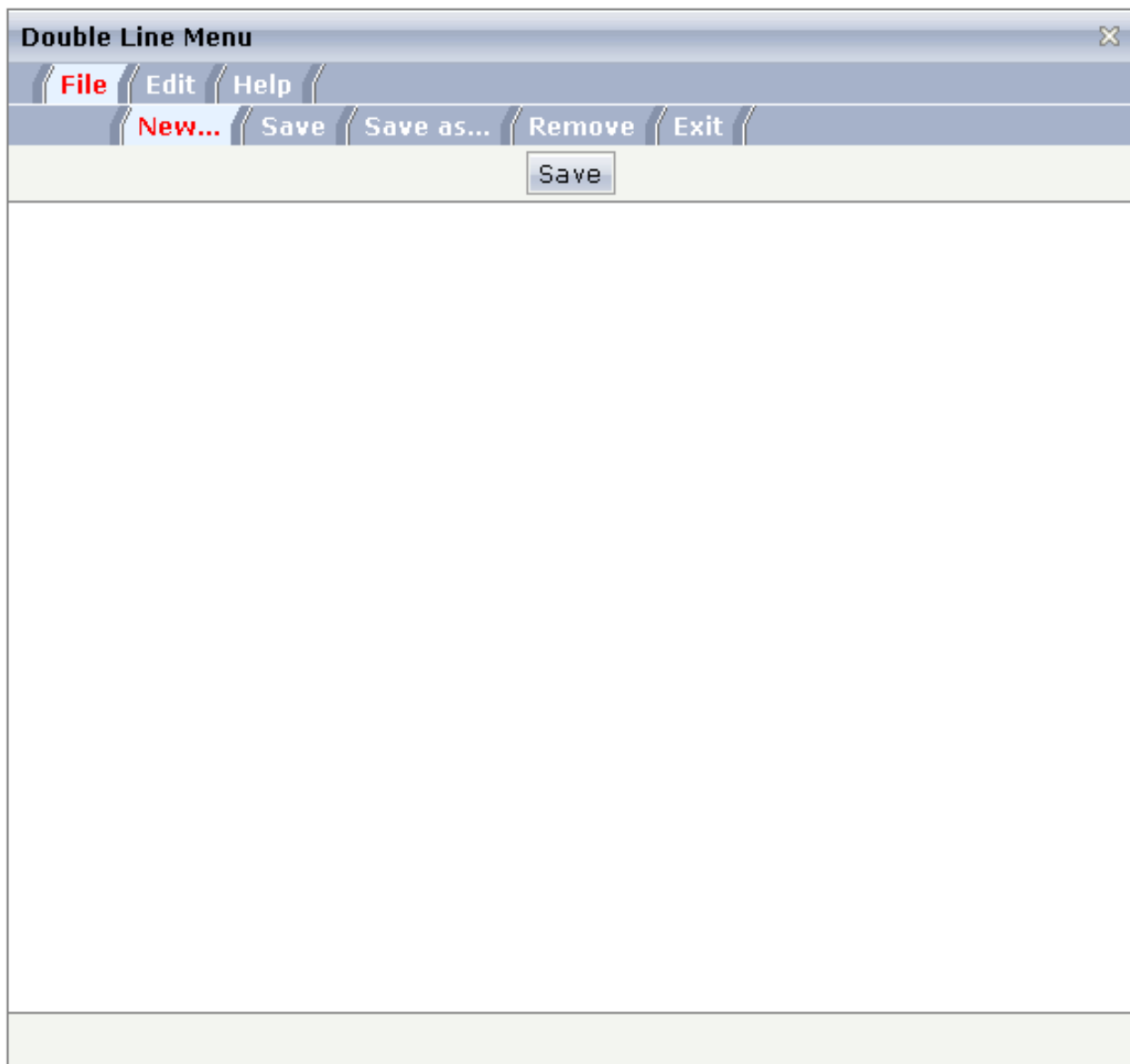
103

DLMENU

▪ Example	694
▪ Adapter Interface	695
▪ Built-in Events	695
▪ Properties	696

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_d1_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

```
DEFINE DATA PARAMETER
1 ITEMS (1:*)
2 LEVEL (I4)
2 METHOD (U) DYNAMIC
2 TEXT (U) DYNAMIC
END-DEFINE
```

Built-in Events

`items.onSelectSubItem`

Properties

Basic			
menuprop	Name of the adapter parameter that represents the control in the adapter.	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	
align	Horizontal alignment of the control's content.	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	

104

XCIPOPUPMENU - Enable Context Menus

■ Example	698
■ Adapter Interface	699
■ Built-in Events	700

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:

- TEXTGRID2
- TEXTGRIDSSS2
- TREENODE2
- CLIENTTREE

The following events are raised when the user right-clicks in corresponding areas of the page:

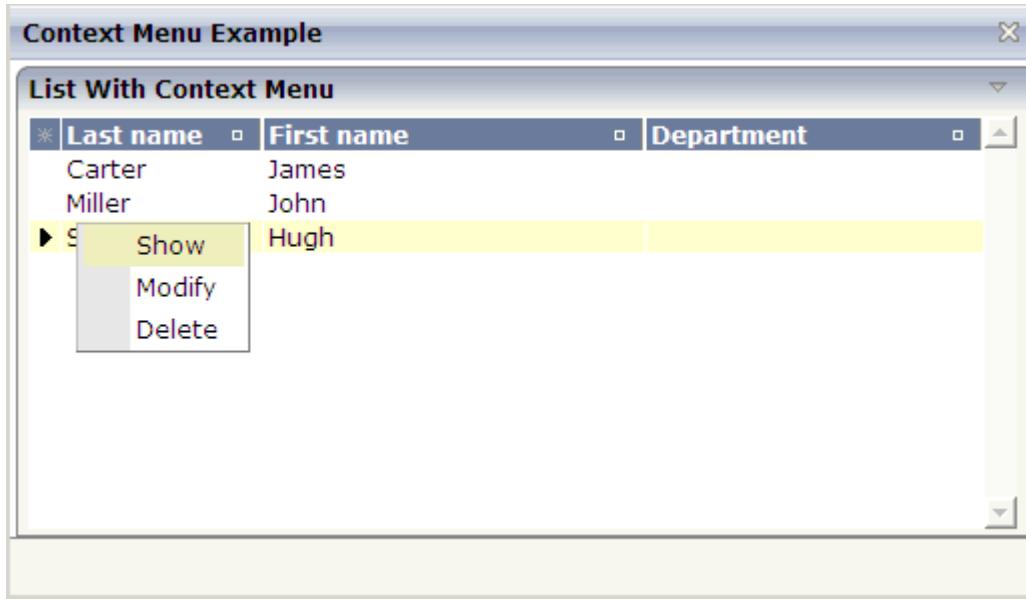
- When the user right-clicks in a non-empty line in a grid or tree, the event *value-of-griddataprop.reactOnContextMenuRequest* or *value-of-treecollectionprop.reactOnContextMenuRequest* is raised.
- When the user right-clicks in an empty line in a grid or tree, the event defined in the property *contextmenumethod* of the grid or tree is raised.
- When the user right-clicks elsewhere in the page, the event defined in the *contextmenumethod* of the page is raised.

In the event handler of these events, you do not 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 library SYSEXNJX as program CTRCTX-P.

Adapter Interface

```
DEFINE DATA PARAMETER
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 ORIGINATORQUERY (A) DYNAMIC
2 SELECTEDREFERENCE (A) DYNAMIC
END-DEFINE
```

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.

The value of `ORIGINATORQUERY` is normally not used by Natural applications. The selected line can easier be determined with an `NJX:EVENTDATA` control.

Built-in Events

`xcipopupmenu.reactOnSelect`

105

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:

- **TIMER**
- **XCIDATADEF - Data Definition**
- **Extended Hot Key Management**
- **Function Key Handling**

106

TIMER

▪ Example	704
▪ Properties	705

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.

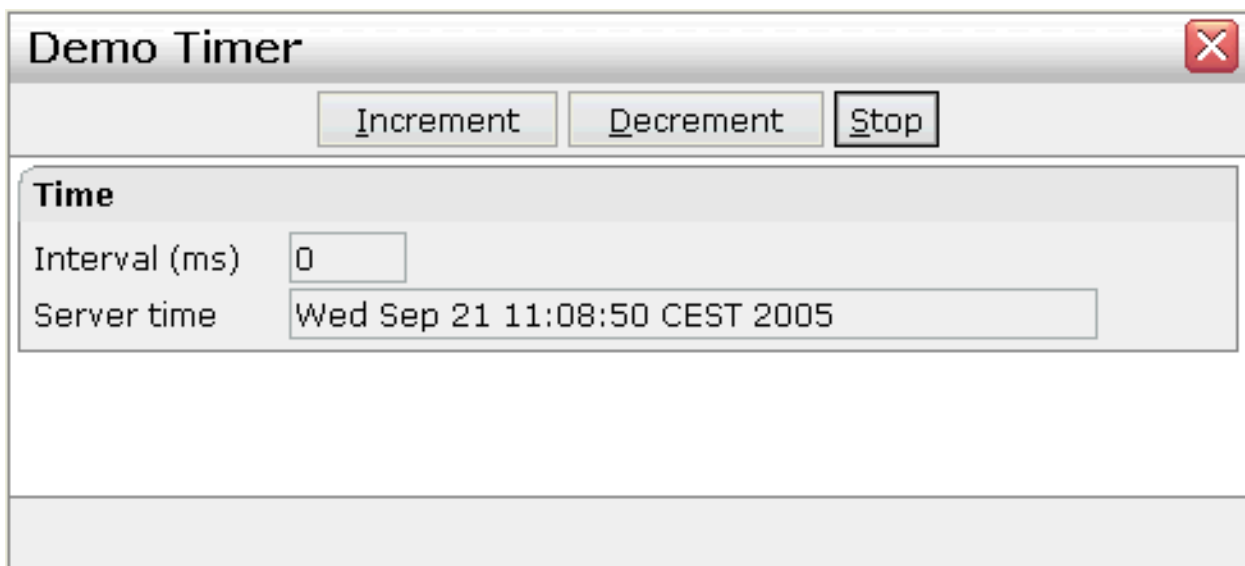
The timer tag is accessible as a valid subnode inside the page tag.

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:

```
<page model="DemoTimerAdapter">
  <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>
</page>
```

```

</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>
</page>

```

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	

107

XCIDATADEF - Data Definition

▪ Example	708
▪ Properties	711

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.

ID	Last Name	First Name
1		
2		
3		
4		

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-page" datatype="C">
</xcidatadef>
<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 library SYSEXNJX as program CTRCV-P.

Properties

Basic			
dataprop	The XCIDATADEF control allows to create data structures for the processing side that are created in addition to data structures that are created by the normal controls. The DATAPROP represents the name of the data element that provides the content of the control.	Optional	
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

			<div>xs:byte</div> <div>xs:short</div> <div>-----</div> <div>N n.n</div> <div>P n.n</div> <div>string n</div> <div>C</div>
array	If set to true, the XCIDATADEF will be an array.	Optional	<div>true</div> <div>false</div>

108

Extended Hot Key Management

■ Direct Hot Key Definitions with Certain Controls	714
■ Hot Key Definitions for Certain Controls	714

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:

Hotkey Enhancements

Info

(1) Within the keycode you can use "textids". With that you can provide for language specific hotkeys - e.g. keys "Ctrl-Alt-d" within an english context may invoke the same functionality like "Ctrl-Alt-l" when you run your application with german language.

(2) You can provide for grid hotkeys. A grid hotkey is is invoked only if the user invokes the hotkeys within the grid.

Demo

Press keys "Ctr-Alt-a" within the grid - a popup will appear. Press the same keys within the surrounding page - the result is now a message on the statusbar.

	First name	Last name
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

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.

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.

109

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.

Exception

In Internet Explorer 7, F10 and F11 are handled by the web browser only if both the keyboard focus and the mouse pointer are in the area of the web browser.

110

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**
- **NJX:XCIWPINFO2**
- **NJX:XCIWPFUNCTIONS**
- **NJX:XCIWPACCESS2**

111

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. See section *Session Management* in *Working with Pages* (which is part of the Application Designer documentation) for more details.

112

Definition of Multi Frame Pages

▪ MFPAGE	724
▪ MFCISFRAME	726
▪ MFHTMLFRAME	728
▪ MFFRAMESET	729

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	<p>Id of the frame. Must be unique inside the frameset page. Must only contain alphanumeric characters.</p> <p>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. <code>openeCIPageInFrame(...)</code>). 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.</p>	Obligatory	
cisurl	<p>URL of the page to be shown inside. Use <code>/project/page.html</code> as syntax, e.g. <code>"/HTMLBasedGUI/empty.html"</code>.</p> <p>Do NOT use only <code>page.html</code> 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!</p>	Obligatory	
comment	Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.	Optional	
Appearance			
resizable	<p>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.</p> <p>Valid values are "true" and "false". Default is "true".</p>	Optional	<p>true</p> <p>false</p>
withborder	Boolean value defining if the frame has a border on its own. Default is "false".	Optional	<p>true</p> <p>false</p>
framestyle	Style that is passed to the HTML-FRAME definition that is internally generated.	Optional	<p>background-color: #FF0000</p> <p>color: #0000FF</p> <p>font-weight: bold</p>

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	<p>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.</p> <p>Do not define this property just "by accident" but leave it to the default ("NOTHING").</p> <p>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.</p>	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. <code>openeCIPageInFrame(...)</code>). 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 <code>"../HTMLBasedGUI/workplace/header2.html"</code> - or <code>"http://www.softwareag.com"</code> .	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	<p>Id of the frame. Must be unique inside the frameset page. Must only contain alphanumeric characters.</p> <p>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. <code>openeCIPageInFrame(...)</code>). 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.</p>	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

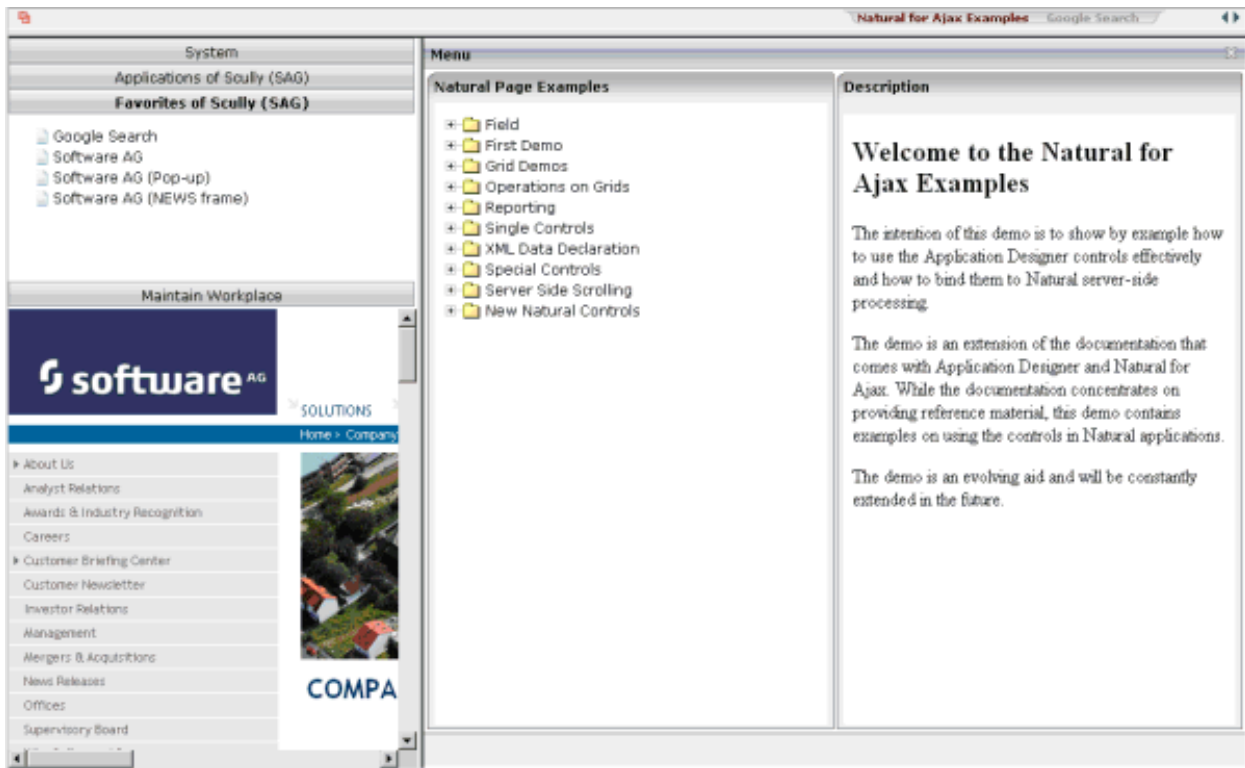
			font-weight: bold
--	--	--	-------------------

■ Framework Overview	735
■ Functions Frame: MFWPFUNCTIONS	736
■ Active Functions Frame: MFWPACTIVEFUNCTIONS	738
■ Content Frame: MFWPCONTENT	739
■ Filling the MFWPFUNCTIONS Frame Initially: MFWPBOOTSTRAPINFO	741
■ Session Management inside the Workplace	750
■ Workplace API for Dynamic Manipulation	751

The Natural example library SYSEXNJX provides 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`

For information on the Natural versions with which this example is provided, see the section *Support for Special Features*.

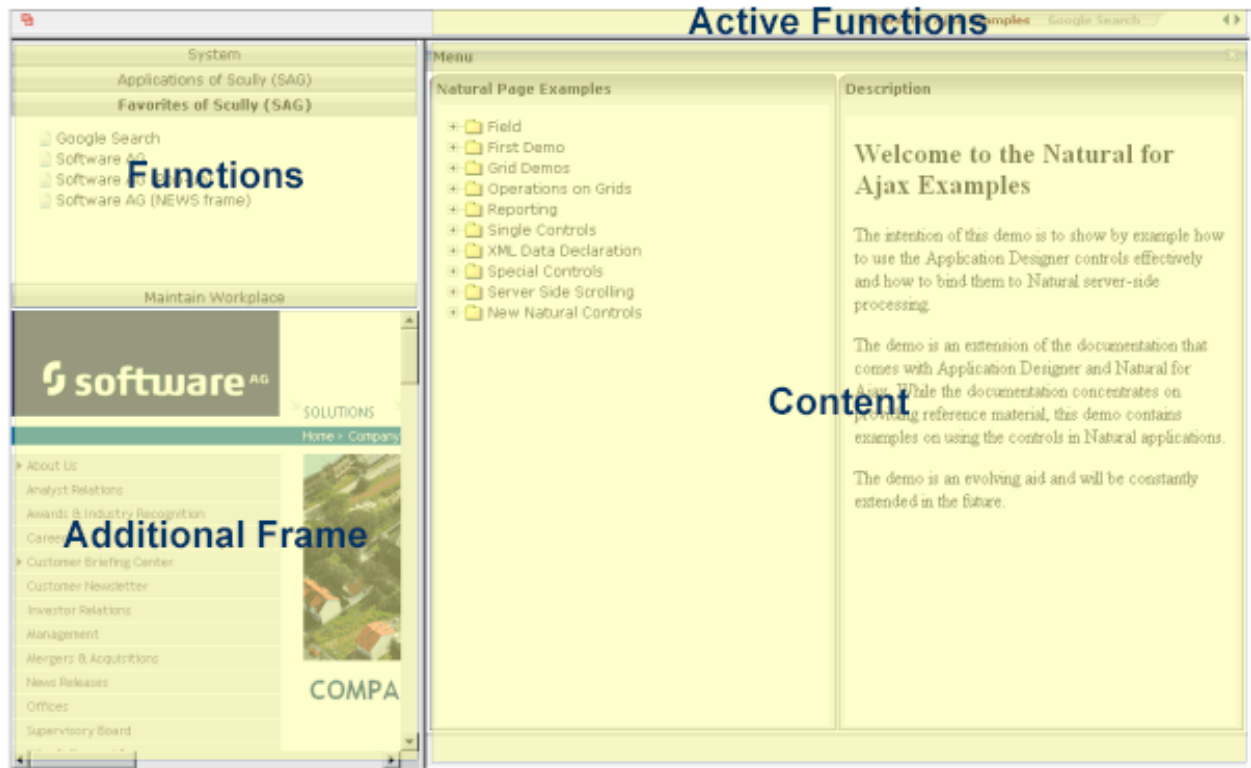


The workplace framework bases on the multi frame page management described in the previous sections. It offers the following:

- 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 (`../nfx<nn>.ear/cisnatural.war/njxdemos/xml/wpdynworkplace.xml`):

```

<?xml version="1.0" encoding="UTF-8"?>
<mfpag 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/html/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>
  </mfpag>

```

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	<p>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.</p> <p>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.</p>	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

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, `../njs<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"
    synchtabsnavigation="true"
    showdustbin="true"
    withtakeouttopopup="false"
    withcloseallwindowsicon="false"

mfworkplaceeventlistener="com.softwareag.cis.workplace.MFDefaultEventListener"
    targetnameofresizableleftpart="AVAILABLEACTIVITIES"
    translationproject="tshmf"
    translationreference="mfworkplace">

    <mfwptopic
        name="System"
        treeclass="WORKPLACETOPIC1ClientTree">

        <mfwpfolder
            name="System"
            draginfo="System"
            opened="true">

            <mfwpopencispage
                name="Login"
                activityurl="/cisnatural/NatLogon.html&xciParameters.natsession=Workplace
&xciParameters.natparam=stack%3D%28LOGON+SYSEXNIX%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&xcParameters.natsession=Workplace
&xcParameters.natparam=stack%3D%28LOGON+SYSEXNJX%3BWPFUNC-P%29"
        onlyoneinstance="true"
        followpageswitches="true">
    </mfwpopencispage>

    <mfwpopencispage
        name="Maintain Content Pages"
        activityurl="/cisnatural/NatLogon.html&xcParameters.natsession=Workplace
&xcParameters.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

```

Each of the sublevel tags can contain all sublevel tags as subnodes, including itself.

The following topics are covered below:

- [MFWPBOOTSTRAPINFO Properties](#)
- [MFWPTOPIC Properties](#)
- [MFWPFOLDER Properties](#)
- [MFWPOPENCISPAGE Properties](#)
- [MFWPOPENCISPOPUP Properties](#)
- [MFWPOPENCISTARGET Properties](#)
- [MFWPOPENHTMLPAGE Properties](#)
- [MFWPOPENHTMLPOPUP Properties](#)

■ MFWOPENHTMLTARGET Properties

MFWPBOOTSTRAPINFO Properties

Basic			
defaultcontentpage	<p>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:</p> <p>(1) Either create one "background page" which always is shown in an "empty" workplace.</p> <p>(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.</p> <p>EXAMPLE: "/HTMLBasedGUI/empty.html"</p>	Optional	
workplacestylesheet	<p>The stlye 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 CISession 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.</p> <p>EXAMPLE: "/cis/styles/XYZ_STLYE.css"</p>	Optional	background-color: #FF0000 color: #0000FF font-weight: bold
translationproject	<p>Name of the project where the actual used multilanguage file is located.</p> <p>e.g. cisdemos</p>	Optional	
translationreference	<p>Name of the multilanguage .csv file.</p> <p>e.g. test</p> <p>(if the file test.csv should be used)</p>	Optional	
comment	<p>Comment without any effect on rendering and behaviour. The comment is shown in the layout editor's tree view.</p>	Optional	
Appearance			
mfworkplaceeventlistener	<p>Use this interface to react on workplace events.</p> <p>(1) Create an implementation of this interface</p> <p>(2) Use method MFWorkplaceInfo.registerMFWorkplaceEventListener to register your class</p>	Optional	

	<p>(3) Use method <code>NODEInfo.setDropInfo</code> on each tree item to be able to drag that item</p> <p>Step two and three are typically done within the "bootstrap info provider"-class</p> <p>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).</p>		
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	<p>Flag that indicates whether the dustbin (have a look at the DEMO WORKPLACE) is shown or not.</p> <p>Boolean value, default is false.</p>	Optional	<p>true</p> <p>false</p>
synctabnavigation	<p>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.</p> <p>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 <code>tabText</code> (<code>setTabText</code>) in order to make nodes unique!</p> <p>true ==> synchronization is done; false ==> synchronization is not done;</p> <p>default is false.</p>	Optional	<p>true</p> <p>false</p>
withcloseallwindowsicon	<p>Flag that indicates whether the CloseAllWindowsIcon is shown in the workplace or not.</p> <p>Boolean value, default is false.</p>	Optional	<p>true</p> <p>false</p>
withtakeouttopopup	Flag that indicates	Optional	<p>true</p> <p>false</p>

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 "andamp;param1=value1andamp;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 id brought back to the starting page always.	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 "andamp;param1=value1andamp;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 "andamp;param1=value1andamp;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	
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	

MFWOPENHTMLPOPUP 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

MFWOPENHTMLTARGET 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	

Session Management inside the Workplace

When the user selects functions in the MFWPFUNCTIONS frame, then pages are opened in the content frame, or as popups 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 popup 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](#)

114

NJX:XCIWPINFO2

▪ Example	754
▪ Adapter Interface	754

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.

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:xciwpinfo2>
    </njx:xciwpinfo2>
  </natpage>
```

Adapter Interface

```
1 XCIWPINFO_CHANGEINDEX (I4)
1 XCIWPINFO_NODE (1:*)
2 ACTIVITYURL (A) DYNAMIC
2 BUTTONSTYLE (A) DYNAMIC
2 DRAGINFO (A) DYNAMIC
2 FOLLOWPAGESWITCHES (L)
2 HEIGHT (I4)
2 ICONURL (A) DYNAMIC
2 LEFT (I4)
2 LEVEL (I4)
2 NAME (A) DYNAMIC
2 ONLYONEINSTANCE (L)
2 OPENED (I4)
2 TARGET (A) DYNAMIC
2 TOOLTIP (A) DYNAMIC
2 TOP (I4)
2 TREECLASS (A) DYNAMIC
2 TREESTYLE (A) DYNAMIC
2 TYPE (A) DYNAMIC
2 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.
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.
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.
OPENED	The following definition means "The folder is closed": OPENED = 0

Element	Meaning
	The following definition means “The folder is opened”: OPENED = 1
TOOLTIP	Text of the tooltip for the folder.

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.
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
NAME	Name of the node.
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.
OPENED	See the above description for LEVEL.
TOOLTIP	Text of the tooltip for the tree node.
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
NAME	Name of the node.
OPENED	See the above description for LEVEL.
TOOLTIP	Text of the tooltip for the tree node.
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

Element	Meaning
	The following definition creates a node on level 3, that is, under a folder: LEVEL = 3
NAME	Name of the node.
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.
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
```

115

NJX:XCIWPFUNCTIONS

▪ Example	760
▪ Adapter Interface	760

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.

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:xciwpfunctions>
    </njx:xciwpfunctions>
</natpage>
```

Adapter Interface

```
1 XCIWPFUNCTIONS (1:*)
2 CMDADDFOLDER
3 ADDFOLDER_ASFIRST (L)
3 ADDFOLDER_FOLDERNAME (A) DYNAMIC
3 ADDFOLDER_OPENED (I4)
3 ADDFOLDER_TOPICNAME (A) DYNAMIC
2 CMDADDNODE
3 ADDNODE_ACTIVITYURL (A) DYNAMIC
3 ADDNODE_ASFIRST (L)
3 ADDNODE_FOLDERNAME (A) DYNAMIC
3 ADDNODE_HEIGHT (I4)
3 ADDNODE_LEFT (I4)
3 ADDNODE_NAME (A) DYNAMIC
3 ADDNODE_TARGET (A) DYNAMIC
3 ADDNODE_TOP (I4)
3 ADDNODE_TOPICNAME (A) DYNAMIC
3 ADDNODE_TYPE (A) DYNAMIC
3 ADDNODE_WIDTH (I4)
2 CMDADDTOPIC
3 ADDTOPIC_SWITCHTOTOPIC (L)
3 ADDTOPIC_TOPICNAME (A) DYNAMIC
3 ADDTOPIC_TREECLASS (A) DYNAMIC
2 CMDREMFOLDER
3 REMFOLDER_FOLDERNAME (A) DYNAMIC
3 REMFOLDER_TOPICNAME (A) DYNAMIC
```

```

2 CMDREMNODE
3 REMNODE_FOLDERNAME (A) DYNAMIC
3 REMNODE_NAME (A) DYNAMIC
3 REMNODE_TOPICNAME (A) DYNAMIC
2 CMDREMTOPIC
3 REMTOPIC_TOPICNAME (A) 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
<code>ADDTOPIC_SWITCHTOTOPIC</code>	"true": Open the new topic.
<code>ADDTOPIC_TOPICNAME</code>	Name of the topic.
<code>ADDTOPIC_TREECLASS</code>	Sets the style class for rendering the tree area of the topic. There are ten standard style classes available in the default style sheet: <code>PLACETOPIC1ClientTree</code> to <code>WORKPLACETOPIC10ClientTree</code> . 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
<code>ADDFOLDER_ASFIRST</code>	"true": Add this folder as the first folder under the given topic. "false": Add this folder as the last folder under the given topic.
<code>ADDFOLDER_FOLDERNAME</code>	Name of the folder.
<code>ADDFOLDER_OPENED</code>	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.
<code>ADDFOLDER_TOPICNAME</code>	Name of the topic to which the folder is to be added.

Add a node

The following structure elements belong to CMDADDNODE:

Element	Meaning
ADDNODE_NAME	Name of the node to be added.
ADDNODE_FOLDERNAME	Name of the folder to which the node is to be added.
ADDNODE_TOPICNAME	Name of the topic that contains this folder.
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_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 element belongs to CMDREMTOPIC:

Element	Meaning
REMTOPIC_TOPICNAME	Name of the topic to be removed.

Remove a folder

The following structure elements belong to CMDREMFOLDER:

Element	Meaning
REMFOLDER_FOLDERNAME	Name of the folder to be removed.
REMFOLDER_TOPICNAME	Name of the topic that contains the folder.

Remove a node

The following structure elements belong to CMDREMNODE:

Element	Meaning
REMNODE_FOLDERNAME	Name of the folder that contains the node to be removed.
REMNODE_NAME	Name of the node to be removed.
REMNODE_TOPICNAME	Name of the topic that contains the folder with the node to be removed.

116

NJX:XCIACCESS2

▪ Example	766
▪ Adapter Interface	766

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.

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:xciwpaccess2>
  </njx:xciwpaccess2>
</natpage>
```

Adapter Interface

```
1 XCIWPACCESS2 (1:*)
2 CMDADDPAGETOWORKPLACE
3 ADD_ACTIVITYURL (A) DYNAMIC
3 ADD_NAME (A) DYNAMIC
3 ADD_TYPE (A) DYNAMIC
2 CMDCLOSECONTENTPAGE (A) DYNAMIC
2 CMDINVOKEMETHODINCONTENTPAGE
3 METHOD (A) DYNAMIC
2 CMDOPENPAGEINTARGET
3 OPEN_ACTIVITYURL (A) DYNAMIC
3 OPEN_TARGET (A) DYNAMIC
3 OPEN_TYPE (A) DYNAMIC
2 CMDOPENPOPUP
3 POPUP_ACTIVITYURL (A) DYNAMIC
3 POPUP_HEIGHT (I4)
3 POPUP_LEFT (I4)
3 POPUP_TITLE (A) DYNAMIC
3 POPUP_TOP (I4)
3 POPUP_TYPE (A) DYNAMIC
3 POPUP_WIDTH (I4)
2 CMDSHOWPAGEINWORKPLACE
3 SHOW_ACTIVITYURL (A) DYNAMIC
3 SHOW_NAME (A) DYNAMIC
3 SHOW_TYPE (A) 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_NAME	The name to be displayed in the “Active Functions” frame.
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_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_NAME	Name of the page in the "Active Functions" frame.
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`.

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.

Index

A

- Ajax, 1
- Application Designer
 - Natural tools for map conversion, 128

C

- conversion logs
 - Natural tool for map conversion, 129
- conversion rules
 - Natural tool for map conversion, 128

M

- map converter
 - Natural tool for map conversion, 128

N

- Natural for Ajax, 1

