

# Casabac GUI Server – Release 2.0

This document summarizes the enhancements of the Casabac GUI Server that are planned for Release 2.0:

- Planned Ship Date for Release 2.0: May 2003
- First 2.0 builds: from March 2003 on
- Compatibility: 100% upwards compatible for 1.\* releases

The specifications are published in order to obtain feedback. Please feel free to send your feedback to [info@casabac.com](mailto:info@casabac.com). Thank you!

**SPECIFICATION „REPORTING“ .....2**

**SPECIFICATION „PERSONALIZATION“ .....5**

**SPECIFICATION „PDF AND PRINTING“ .....9**

**SPECIFICATION „PERFORMANCE“ .....13**

**SPECIFICATION „DEVELOPMENT TOOLS“ .....14**

**SPECIFICATION „CONTROLS“ .....15**

The specifications tell what we are planning to do – Casabac does not guarantee that all Release 2.0 features will be implemented exactly the way they are described within this document.

# Specification „Reporting“

## Release

- Casabac GUI Server 2.0

## Goal

- Provide for simple but flexible way to develop typical reporting pages for querying contents of a relational database
- Typical query screens consist of
  - Input of selection criteria
  - Output of query result as grid
- In the grid output navigation possibilities can be added.
- The columns offered in a grid can be selected at runtime.
- Any kind of accumulation and calculation can be done inside the grid.

While the reporting functions should on the one hand provide for a fast way to create standard report screens the concept itself should be open: reports are not “closed applications” that are somehow generated and must not be touched by developers afterwards – but just the opposite: via reporting functions developers can use generic ways to define their screens but still have the possibility to add their application specific code.

## Solution – Generic DB Controls

### Database Controls

Casabac offers a set of controls that simplify programming against the database. The controls can be used in order to build up reporting screens using the normal Casabac Layout Painter in a fast and efficient way.

The controls are:

- DBFIELD
- DBCOMBO
- DBRADIOBUTTON
- DBCHECKBOX
- DBCONDITION
- DBTEXTGRID

Based on the controls a wizard is offered to build standard reporting screens without any explicit implementation work required.

### DBFIELD Control

The DBFIELD Control is taken as example to explain the principle behind. The following definition may be done within a layout:

```
<dbfield valueprop=' department '  
width=' 200'  
datasource=' standard '  
reftable=' DEPARTMENTS'
```

```

    refcolumn=' ID'
    refdescrcolumn=' TEXT' >
</dbfield>

```

At runtime this definition automatically creates a field on the screen that provides for the following functions:

- Automated "Find Valid Values"
- Automated check of manual input against database

All the "normal coding" needs not to be done in more. The server side processing of the control just expects a connection from the data source "standard". The connection is provided for by an interface that the adapter has to implement.

DBCOMBO, DBRADIOBUTTON, DBCHECKBOX are built in a similar way.

## DBCONDITION Control

The DBCONDITION control allows the input of several query conditions for one column. The result is transferred into a proper query condition on server side.

## DBTEXTGRID Control

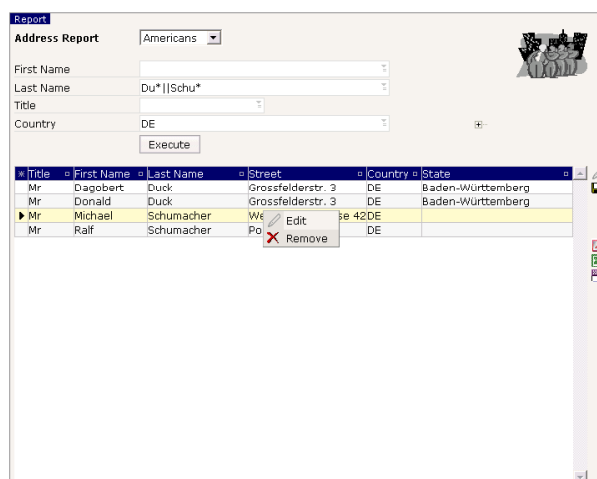
The DBTEXTGRID Control is a TEXTGRID that you can pass a JDBC result set of an SQL query.

## Solution – Extended DBQuery Control

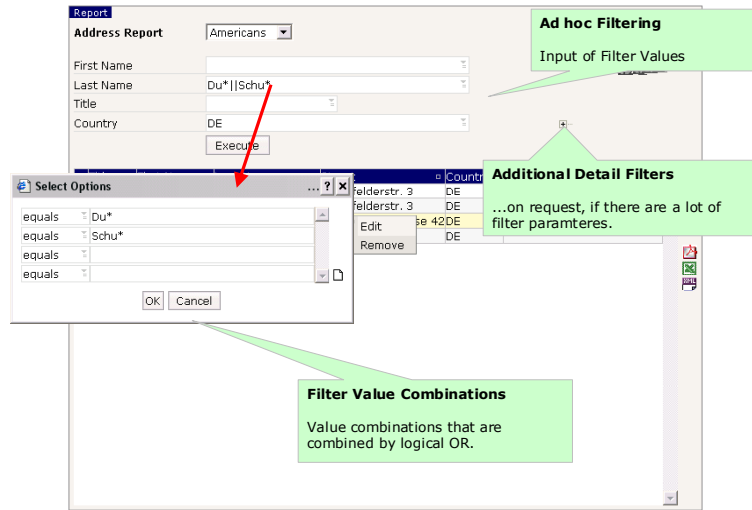
On base of the controls mentioned in the previous controls a combined control is provided for that simplifies the creation of standard list reports. The definition of the control includes:

- Logical naming of data source
- SELECT statement against DB to be executed
- Definition of filter criteria
- Definition of output columns

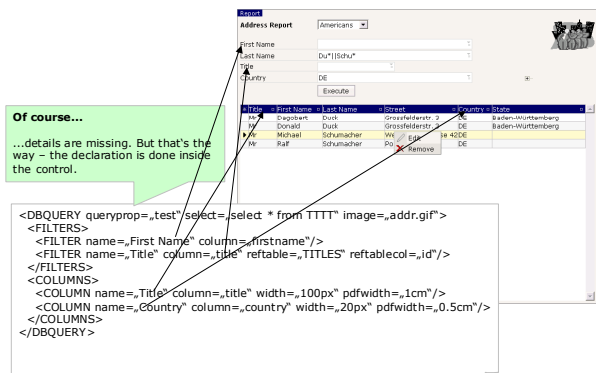
All the steps in between are done automatically. The following images give an impression about the look and feel and planned capabilities:



The control allows various filtering options. Filters may have connection to reference tables, i.e. a value help based on DB references is automatically provided for.



The control is completely configured inside the Layout Painter of Casabac. There is no need – if not explicitly required - for server side programming on application side besides providing for a connection.



The server side application can connect to the control via various interfaces. If a item of the selection result is chosen then the application defines the popup menu to be shown and can individually react on function selection.

## Not covered

- Any kind of extended graphical reporting
- Any kind of document output reporting ("Small Crystal Reports")

# Specification „Personalization“

## Release

- Casabac GUI Server 2.0

## Goal

(1) Have customer specific screens based on standard screens:

- Customer specific screens are “like” standard screens but with modifications
- Types of modification
  - Remove control
  - Switch control to display only
  - Exchange single attributes of a control (e.g. style, text, ...) which are not relevant for the adapter processing

(2) Changes to the screen are to be done in front of the adapter logic. The same adapter should work with the personalized screens.

(3) Changes are applied one time only. If the standard screen is updated then the changes to the personalized screens are re-done in an automated way.

(4) Possibility to define “Personalization Sequences”:

- Have multiple personalized screens for one standard screen. E.g.:
  - Standard screen
  - Business Unit personalization
  - Organization personalization
- Personalized screens are to be chosen along a certain access sequence: if there is no organization specific screen then the business unit screen is used. If there is no company screen then the standard screen is used.

(5) Have an easy way to build personalized screens.

- Optimal: Layout is displayed. Customer selects in layout control and defines the personalization on the control.

## Solution

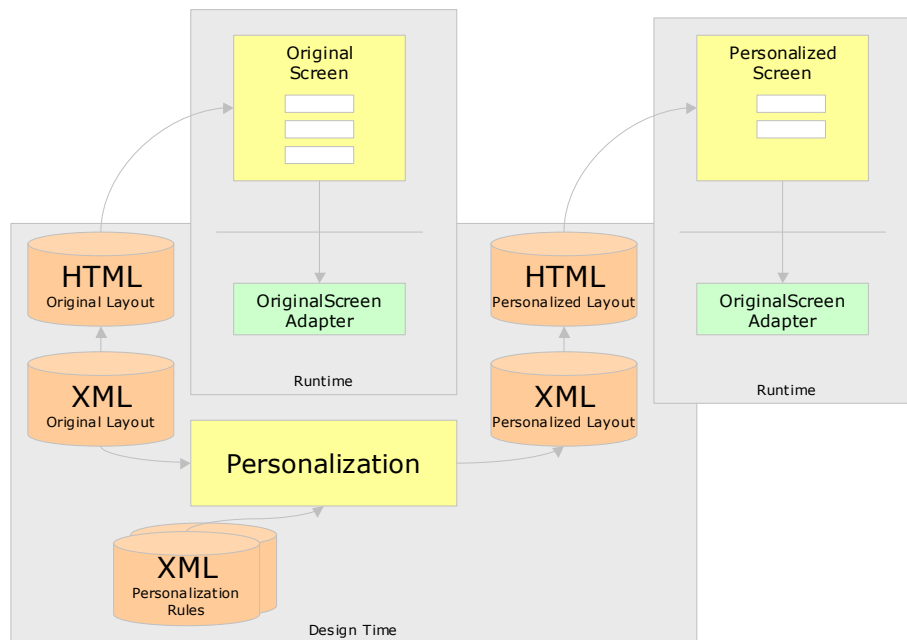
### Concept

The customer can maintain an own set of pages instead of original pages. The pages are normal Casabac pages that are created by taking an original pages and applying some conversions on. The pages refer to the same adapter as the original page.

Consequence:

- No change in adapter logic necessary
- Personalization happens “in front of adapter” and needs not to be implemented inside your adapter logic.

Personalized pages are derived from original pages by defining "personalization rules":



By explicitly keeping personalization rules the personalization can be reapplied to pages after during application release changes. The customer needs not to modify all screens again but just need to reapply all rules.

## Personalization Definition

Per screen one or more personalization definitions can be specified. A personalization definition is stored within an XML file below the project's personalization directory. Example

```
..... /webapps/CasabacGUI Server/casabacdemos/personalization/screenxyz.PERS_ABC.xml
```

This file keeps the personalization settings for screen "screenXYZ.xml" for personalization scenario "ABC".

Inside the file controls of the original screen are referenced – together with a certain activity that is to be performed:

```
<personalization>
  <control type="field"
    attribute1="valueprop" value1="firstName">
    <replace replacetype="hdist"
      attribute1="width" value1="STAKEOVERS"
      attribute2="style" value2="background color: #FF0000">
    </replace>
  </control>
  <control type="itr"
    containertype="field"
    containngattribute1="valueprop" containngvalue1="country">
    <remove>
    </remove>
  </control>
  <control type="field"
    attribute1="valueprop" value1="zipCode">
    <overwrite attribute1="displayonly" value="true">
    </overwrite>
  </control>
</personalization>
```

The file contains:

- Definition of the control on which a certain activity should happen. The control is referenced in two ways:
  - Control is directly referenced by its type and some of its attributes.
  - Control is referenced by specifying one contained control. This is the typical way of referencing to container controls.
- Definition of the activity that is performed on the control. There are the following activities:
  - Replacement of the control by another one (typically replacement by hdist-control)
  - Optical removing of control ⇒ the control is not transferred into HTML during page generation; it is kept inside the XML file as control with a prefix "removed\_".
  - Overwriting of certain attributes of the control.

## Generation of personalized Pages

Based on the definition of the personalization definition the personalization can be executed. This means that a program generates personalized pages based on the original page and the personalization settings.

Example: if the original page is "xml/screenxyz.xml" and the personalization file is "personalization/screenxyz.PERS\_ABC.xml" then the XML page definition of the screen is "xml/screenxyz.PERS\_ABC.xml".

The generation can be done on project base for all contained page definitions.

## Runtime Context

An application may set a personalization context at runtime (e.g. after a user logged on). The definition is kept on session context level. It can be done in two ways:

- Setting one personalization context:

```
Model.m_sessionContext.setPersonalizationContext("ABC");
```

- Setting a sequence of personalization contexts:

```
Model.m_sessionContext.setPersonalizationContext("ABC_DEPARTMENT", "ABC_COMPANY", "ABC");
```

The Casabac runtime environment dynamically follows the context settings in the following operations:

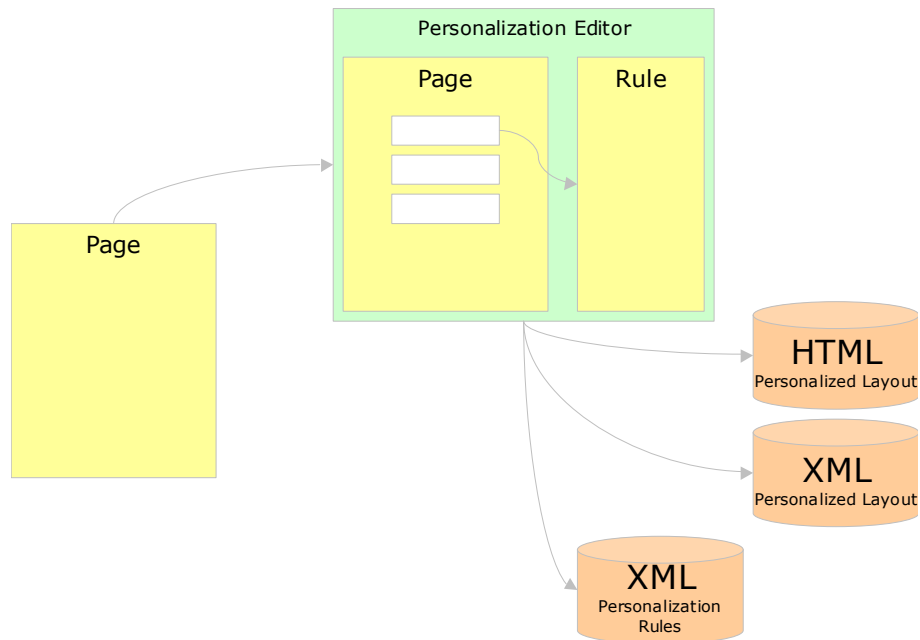
- Call from page through workplace
- switchToPage
- openPopup
- openCasabacPageInTarget

The Casabac runtime first always tries to find a personalized page before – if not existing – uses the standard page. In case of a sequence definition the whole sequence is processed in order to try to find the best fitting page.

## Tools

### Maintenance of Personalization Definition

The personalization definition files can be either created directly or by using a certain tool environment. The "end scenario" looks as follows:



- An administrator user logs on in a certain mode that enables him to define personalization files.
- In a certain page the user presses a personalization function via a hidden key ("ctrl-alt-P"...)
- A personalization editor pops up, The personalization editor looks similar to the current Layout Painter. Inside the personalization editor the user defines the rules as described in the previous chapters.
  - User selects a certain control.
  - Window pops up in which the user can tell which rule to apply
  - Input of rule and parameters of rule
  - Save ==> when saving the existing personalization scenarios are shown; user selects one and saves
  - During save the corresponding XML and HTML files are created and saved.

## Mass Generation

Mass generation functions are offered in able to regenerate all personalized pages for all scenarios in one step. Typical usage: after a release change (of your application) all personalized pages are regenerated once again in order to adapt to your product improvements.

# Specification „PDF and Printing“

## Release

- Casabac GUI Server 2.0
- CHANGE: we in the meantime added a FOP-based solution for printing which is 100% JAVA; FOP will be the base for our pdf and printing strategy

## Goal

- Simple but flexible and efficient form processor that is used for...
  - PDF generation
  - printing documents
- Form processor
  - Define form in text processor (e.g. Word)
  - Add variables into form that bind to data objects (the same way controls are bound inside the GUI Server's layout definition)
  - Merge data object and form at runtime
- PDF Generation
  - Transfer form into PDF via "Print to Postscript" and "Ghostscript"
- Printing
  - Client side printing: user views form as pdf document and can print from the client
  - Server side printing: application can pass form to printer – actual printing via text processor automation

## Solution via FOP Services

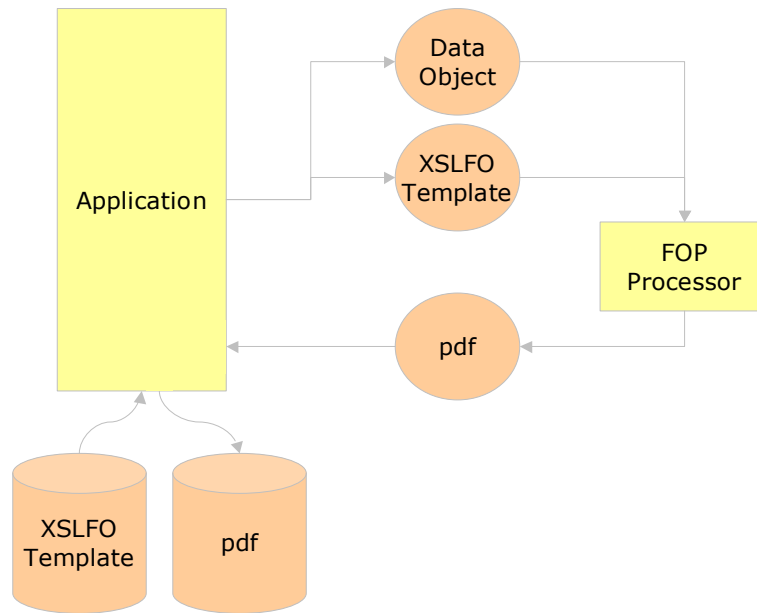
### FOP

FOP stands for "Formatting Objects Processor". This is an Apache project providing for XML based functions to create various output formats – such as pdf – from XML based documents. It is running under GNU license, i.e. usage is without cost.

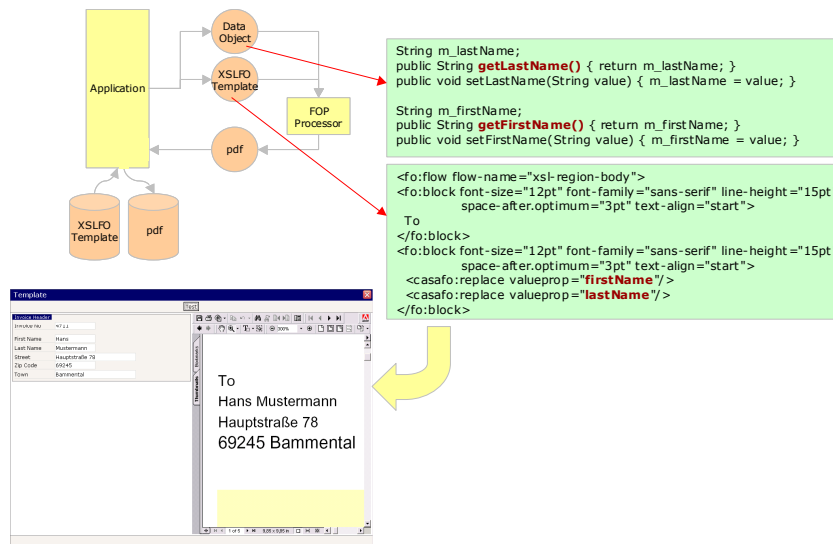
Integral part of FOP is the so called XSLFO – a definition of XML tags to easily define documents that are transferred to pdf afterwards. With XSLFO you can define document layout and content.

### Overview

The following image shows the way how pdf files are created in a simple way from Java:



An XSLFO template is defined. It is mixed with a Data Object and sent to the FOP processor. The preprocessor does a preparising of the template and detects certain extended XML tags. These XML tags are replaced by XSLFO-XML tags. The result is passed to the normal FOP functions transferring the XSLFO-XML stream into pdf.



In the picture you see how the XSLFO template references to values of the data object.

## Details

How does the application provide for the XSLFO template?

- This is completely up to the application. Either the XSLFO template is read from the file system (or database) or it is dynamically created. The FOP processor receives a String as input parameter and passes back a byte-Array containing the pdf.

Which special tags are available containing a special binding to the data object's content?

- Replacements
- Flexible Tables

How is the XSLFO template edited?

- Currently: via a text/XML editor
- Maybe we will provide for some more sophisticated editor, but this is not yet decided.

Is this 100% Java?

- It is.

Which version of FOP will be used?

- In principal: the one that is the newest released one.
- Currently: 0.2.0.5

## Solution via RTF Services

### “Data-RTF” as Form Definition Language

RTF is used as base language for defining the layout. Reasons:

- RTF is a specified format.
- RTF is based on sequential ASCII.
- RTF is supported by “all” text processors.
- RTF is easy to handle.

Other solutions were discussed but not taken into consideration:

- HTML – Is not usable for forms because it is not print-page oriented.
- PDF – Direct generation of PDF is difficult. Lack of available form definition tools on PDF base.

Inside RTF data tags are placed which define...

- what output to render
- what data to render (binding information)

The data tags are identified in the following way:

```
... @@text: valueprop=firstName@@...
... @@table: arrayprop=lines| | column: prop=firstName|width=1000| | column: prop=lastName|width=1000|shading=100@@...
```

### “Data-Tags”

Data Tags are passed to data tag handlers which replace at runtime the data tag with corresponding RTF code. The integration of data tag handlers is similar to the integration of control handlers within the GUI framework.

The following data-tags are supported as standard:

- text
- table
- image

## WinWord Automation as Text Processor

WinWord 2000 is used as text processor for form definition and form output. It serves the following tasks:

- Printing RTF into Postscript via a corresponding printer definition.
- Printing RTF to any printer.

WinWord is called by a "small" Java program via its command line interface. Inside Winword "small" macros are responsible for starting the output to postscript/ printer.

## Text Processor decoupled from JAVA Process

The "small" Java program can be decoupled and can be called via RMI – this means the core Java processing can run on any operation system – just the printing needs to be based on a windows machine.

## Dicussion

Pros

- Simple "WYSIWYG" form processor
- Open - following the same principles as the GUI Server layout definition

Cons

- Dependency on text processor as tool for converting forms into PDF and for printing – platform dependency (though decoupled via a remote interface)

# Specification „Performance“

## Release

- Casabac GUI Server 2.0

## Continued Reduction of exchanged Data Volume

- Goal: reduce data volume by 30%!
- Use available technologies for reducing ing the Javascript files:
  - Take out spaces
  - Take out comments
  - Shorter names for generated variables
  - Only transfer these JS libraries which are required by the page

# Specification „Development Tools“

## Release

- Casabac GUI Server 2.0

## Goal

- Improve the usability of the existing development tools.

## Solution

### Layout Painter

- Select multiple items inside the tree to cut/paste/...
- Online help for controls
- Better Value help for attribute input
  - Introspection of properties(methods for binding definitions)

### Code Generator

- No meaningful comments that are generated into the text anymore ⇒ proper parsing of Java Code

### Literal Translator

-

# Specification „Controls“

## Release

- Casabac GUI Server 2.0

## Control “Multiple Assignment”

The controls looks as follows:

Item 1		Item 5
Item 2		Item 6
Item 3	ç	
Item 4	è	
Item 5		
Item 6		

Items can be easily shifted from the “left” to the “right”.

## ROWTABLE-Grid with dynamic Number of Rows

Currently the number of lines is defined within the control definition. It will now be possible to define the height e.g. as percentage value and let the system dynamically define the number of lines to be rendered.

## Database Controls

A set of new controls that simplify the programming of SQL database based applications is described in the specification for “Reporting”.