Inspire Designer R10

Getting Started

Version 10.0

CLASSIFICATION: PUBLIC

GMC Software AG

Inspire Designer

Getting Started

Product version 10.0

Document version 10.0.0.2

Release date: May 2015

GMC Software AG

www.gmc.net

See also all GMC documentation online

Should you have any queries, suggestions or comments concerning these materials, please do not hesitate to contact us at documentation@gmc.net.

This document contains information classified as Public.

The GMC Software Group of companies are ISO 9001:2008 certified.

Copyright

Information contained within this document may contain technical inaccuracies or typographical errors. Changes will be added periodically and modifications will be made thereto without prior notification. GMC Software AG does not enter into any obligations or responsibilities regarding the content of this document and does not assume any legal liability – neither expressed or implied – for its accuracy, completeness and/or usefulness.

Copying of the software or manual on to any data storage medium or in any other way, except for explicit company internal use, is strictly forbidden without the prior written authorization of GMC Software AG. Failure to comply with these restrictions is liable to prosecution.

Trademarks

GMC Software Technology and its logo are trademarks and service marks of GMC Software AG registered in Switzerland, the US and numerous other countries.

Adobe and Adobe ® PDF Library™ are trademarks or registered trademarks of Adobe Systems Inc. in the US and other countries. The names of other companies, products and services are the property of their respective owners.

Table of Contents

1 Introduction

Inspire Designer is a document design, proof and production software suite that provides unrivalled variable-data printing capabilities.

Inspire Designer provides:

- Variable-data document design and production tools for high-volume printing.
- Advanced job management.
- A true GUI environment.
- Universal platform, standards and protocol support.

Inspire Designer has a broad spectrum of features and finding all of those which suit your needs may require some initial guidance. The aim of this document is to introduce functionality that is useful to every-day variable-data document creation. We will continue through this guide in a progressive way, first introducing the environment, then walking you through some basic practical steps of document creation and, finally, creating a reasonably complex design that, we hope, will allow you the confidence to leave our examples aside and start creating a document exactly how you want it.

2 Key

In the text of this guide, there are some common text indicators that show that a word or paragraph has some specific relevance to either the software being described or the document itself:

- Link [4] – A link to another part of the document.
- *Glossary Term* – A word that is described in the glossary at the end of the guide, with a link to that description.
- **Option** – A name of an attribute of Inspire Designer, e.g. the name of a combo-box for a parameter or a menu item that can be selected.
- *Value* – A value or type in Inspire Designer that can be selected or entered, e.g. in a combo or edit-box.
- ⌖ – The cross-hair icon is used in the Examples chapter to identify the aim of each and every step.
- Remember – A link to previous and closely related explanations.

In the interest of brevity, only the features and attributes that are used in the featured examples will be described. A description of all Inspire Designer features can be found in the help file. Go to the menu **Help | Contents** or use the **Help** tool ? available from the top-right of the application's title-bar.

3 Overview

The GUI of Inspire Designer is made up of a main application window, inside which tabs can be opened. Each tab is an environment geared to a different stage of document creation. These environments can be summarized as:

- Workflow – A workflow is a file that describes a document's composition, i.e. what data and sheets will be used. All of the aspects of a workflow are managed from the Workflow environment. It features both the Workflow window (an environment that graphically represents the components of a workflow's composition) and the Layout window (a powerful editor in which the document, that will be produced when the workflow is run, can be designed).

- Proof – The Proof Tab features the Proof window (an environment that can run a virtual print job from the workflow file) and the Data-Proof window (an environment which can view the variable-data that has been processed by the workflow).

- Production – The Production Tab features the Main Production window and associated windows (Jobs, Drivers, Multiple Jobs) for controlling the printing process for any completed workflow file. A Simple Production environment also exists.

Each of these environments are independent and can be used as if it was a separate application. This considers that not everybody involved in the document creation process will always be interested in all stages involved and so, when Inspire Designer is started, the Startup dialog contains buttons that offer direct access to a selection of the windows mentioned above:

Workflow			Proof	Production	
New Workflow	Workflow Wizard	Layout Window in New Workflow	Proof Environment	Production Environment	Simple Production Environment

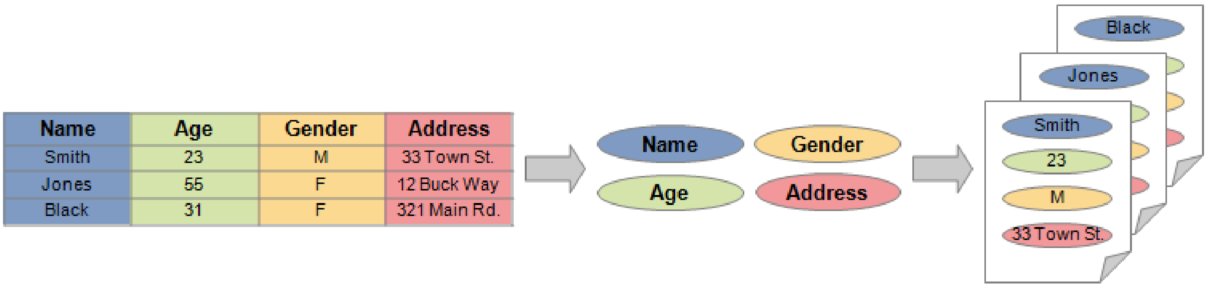
For the purpose of this document, we will approach Inspire Designer with an interest in all of the stages required to create a variable-data document from concept to print, i.e. we will not introduce every window but just those needed to get the job done. This Overview chapter introduces those windows and the Working with Inspire Designer chapter explains how to work within them.

3.1 Workflow

All of the features of a document's composition are defined and brought together in a single WFD file, called a workflow. In general, the function of a workflow is to:

1. Read data files and determine how the data within them should be read into the workflow.
2. Perform required manipulations of the data that has been read into the workflow.
3. Specify the design of the sheets to be printed.
4. Regulate how the pages of the design should be collated.
5. Define the intended printing device and media.

The nature of this process requires that, when the workflow is processed, one document design is printed for each record of the data input. E.g. one designed page is repeatedly printed for all of the data records in a data file:



Name	Age	Gender	Address
Smith	23	M	33 Town St.
Jones	55	F	12 Buck Way
Black	31	F	321 Main Rd.

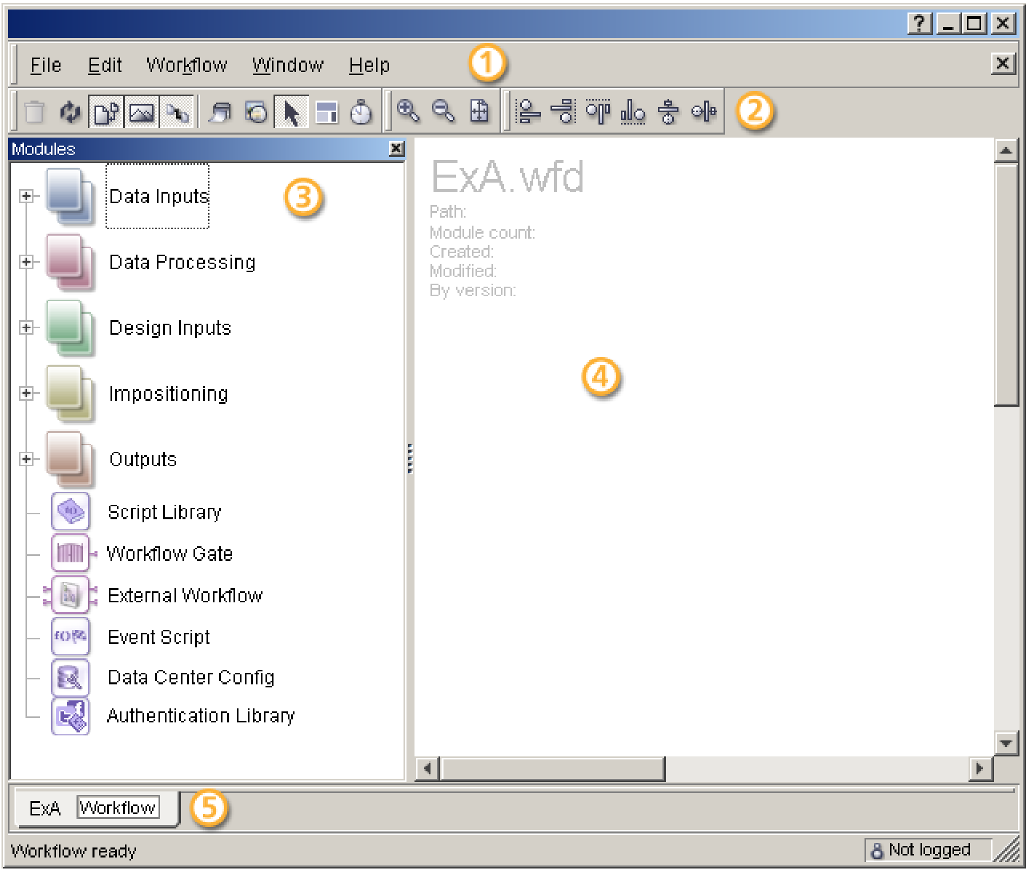
Figure 3.1 The Transfer of Data from Data File to Document Design

However, which records reach the document design and how that design is repeated, are also parameters that can be controlled by the workflow.

All of the tools that may be needed to create a workflow, are collected together in the environment that is represented by a single tab for each workflow opened: the Workflow window.

3.1.1 Workflow Window

The Workflow window is the main environment for creating a WFD file. Workflow creation always begins here.



ExA.wfd
Path:
Module count:
Created:
Modified:
By version:

Workflow ready

Figure 3.2 The Workflow Window

The figure above shows the Workflow window before any workflow design has begun, i.e. a new and unsaved workflow. Introducing the marked features:

1. Workflow Menu
2. Workflow Toolbar

3. Module Tree
4. Workflow Area
5. Tab

The Workflow menu (1) provides access to general settings, other workflow files, windows and window-panes. Editing options for a workflow under construction can be actioned from there and by the icons found in the Workflow Toolbar (2).

The tab (5) at the bottom of the window is a feature common to all of the functional environments of Inspire Designer, it lists all open windows within that environment and all other open tabs are shown along side, allowing navigation between windows and environments.

The Module Tree (3) contains a list of all of the possible workflow components, split into categories using a dependency tree format. The Workflow Area (4) is where combinations of these components can be placed to build a workflow.

The purpose of the Workflow window is to build a workflow with modules. Doing so is described in the following section.

3.1.1.1 Modules

The details that define a workflow, from required input to required output, are the attributes of components. These components are called modules. Modules are graphically represented in the GUI.

Workflow building involves selecting required modules from the Module Tree and placing them in their logical order across the Workflow Area, connecting each module to the others and then configuring it.

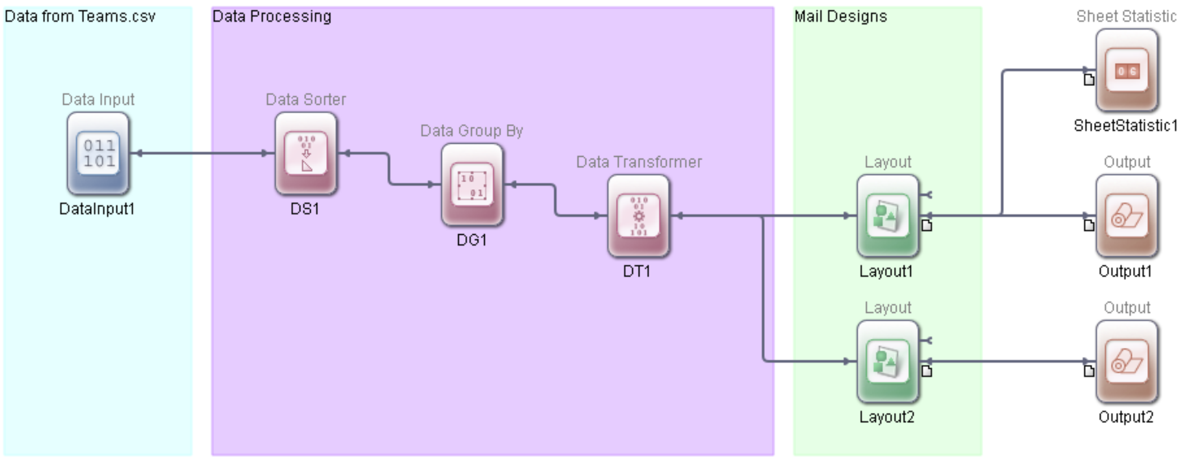


Figure 3.3 An Example Workflow Area – with Workflow Modules Placed and Connected

The range of modules is categorized by their function, which is color coded:

1. Data Inputs
2. Data Processing

3. Design Inputs
4. Impositioning
5. Outputs

Data Inputs (1) read data from data files to create data variables. These are then manipulated by Data Processing modules (2) to achieve a suitable set of variables for a document's design. These variables are then arranged in a document design by a Design Input module (3).

In the next stage of the workflow the sheets of the document can be arranged in preparation for a print medium by impositioning modules (4). Certain parameters for medium selection and the intended printing device are then defined in the Output module (5).

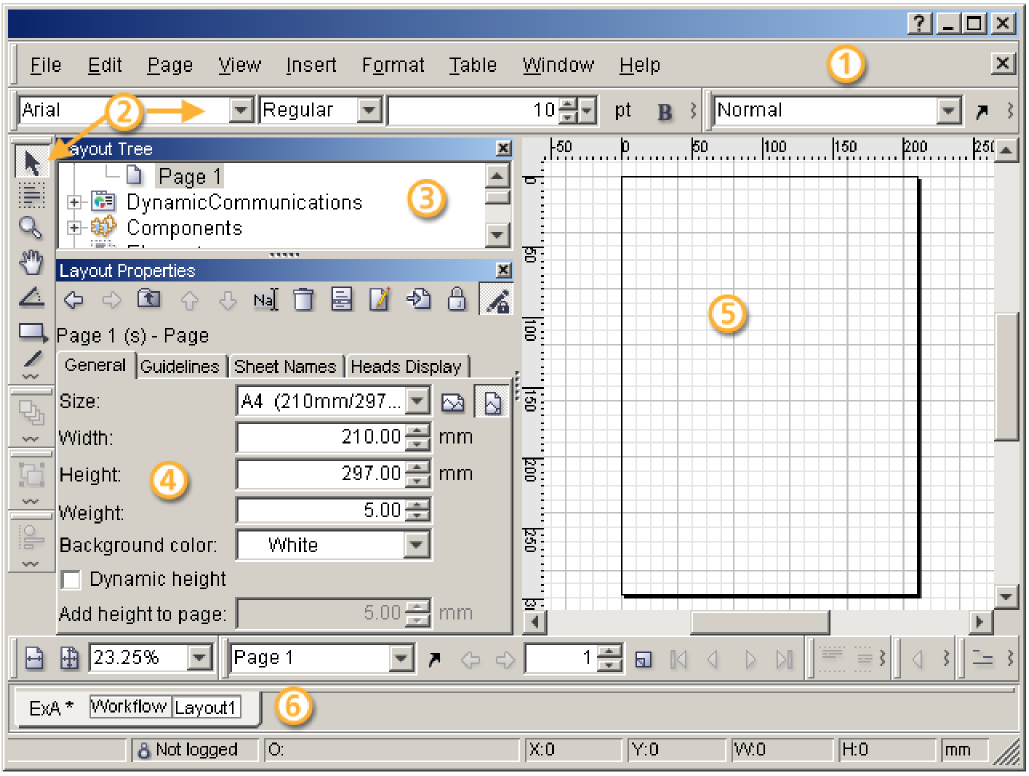
Most modules are configurable and, once placed in the Workflow Area, can be opened so that parameters can be set.

Modular composition is flexible, allowing gradual configuration and focus on the requirements of one part of the workflow at a time. Further, it allows the addition or adjustment of modules as the overall design requirements become more complex. To learn how to work with modules in the Workflow window, go to the Working with Inspire Designer chapter.

The configuration of each module is specific to that module and so, in this guide, we will only introduce how to configure specific modules in context, i.e. if we use them in the Examples chapter. However, the Layout module (one of the Design Inputs) is the workflow's page editor and, as such, requires its own window with many features that need introduction. See the following section for details.

3.1.2 Layout Module

The Layout window is used for designing the pages that the document will consist of. Because it is so commonly used, the Layout window is accessible via the workflow tab.



Arial | Regular | 10 pt | Normal

Page 1 (s) - Page

Size: A4 (210mm/297...

Width: 210.00 mm

Height: 297.00 mm

Weight: 5.00

Background color: White

□ Dynamic height

Add height to page: 5.00 mm

23.25% | Page 1 | 1

Not logged | O: | X:0 | Y:0 | W:0 | H:0 | mm

Figure 3.4 The Layout Window

The figure above shows the Layout window before any document design has begun, i.e. a new page. The basic features are marked:

1. Layout Menu
2. Layout Toolbars
3. Layout Tree
4. Layout Properties Panel
5. Layout Area
6. Workflow Tab

The Layout Area (5) is the workspace for the design. It is an area in which objects can be placed and arranged to create the appearance of a document. A selected object will be shown and can be edited here.

The Layout Tree (3) is a navigable index of all of the objects that are in the design. Its dependency tree form reflects the physical hierarchy of the objects within the WFD file and it is helpful to think of a design in tree form in order to optimize the workflow.

By selecting an object, either in the Layout Tree (3) or Layout Area (5), the Layout Properties panel (4) displays that object's attributes. Each object has its own properties panel, which is commonly a set of tabs that contains combo and edit-boxes so that parameters can be adjusted.

The Layout Menu (1) provides access to general settings, other workflow files and window-panes. The editing options for a design can be actioned from there and by the tools found in the Layout Toolbars (2).

The Workflow tab (6) shows the open windows within the environment. There can be more than one Layout module used in a workflow and, therefore, more Layout windows open on the Workflow tab.

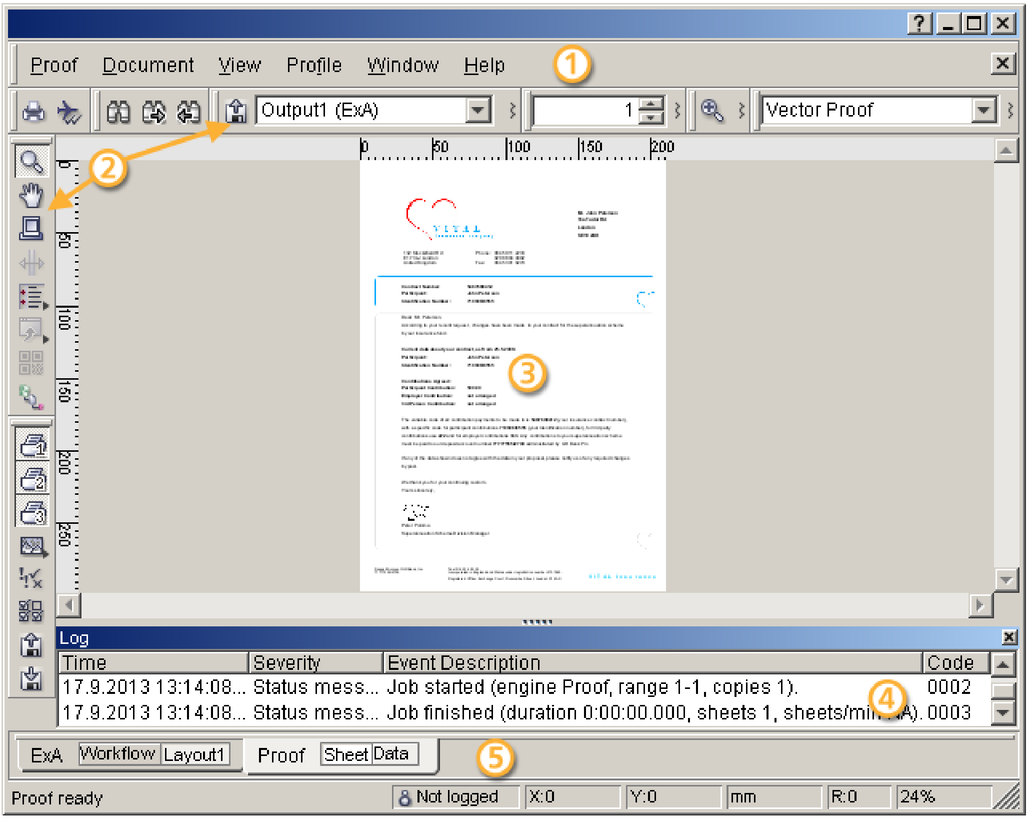
To learn how to work in the Layout window, go to the Working with Inspire Designer chapter.

3.2 Proof

The Proof environment of Inspire Designer provides a preview of how a workflow will look and what its contents will be when it is printed. It allows potential problems to be identified and eliminated before reaching the production phase. It is strongly advised to run Proof during the design stages of any workflow.

3.2.1 Sheet Proof Window

The Sheet Proof window provides a WYSIWYG preview of how the workflow design would look should it be sent to a printer at its present stage. Alternatively, it will warn you if there is an error in the design.



```
Time                Severity       Event Description                                              Code
17.9.2013 13:14:08... Status mess... Job started (engine Proof, range 1-1, copies 1).              0002
17.9.2013 13:14:08... Status mess... Job finished (duration 0:00:00.000, sheets 1, sheets/mi[illegible]A). 0003
```

Figure 3.5 The Proof Window – with an Example Workflow already Loaded

Introducing the basic Sheet Proof window features as they are marked in the figure above:

1. Sheet Proof Menu
2. Sheet Proof Toolbars
3. Sheet Proof Area
4. Log Window-Pane
5. Proof Tab

When a workflow is loaded to proof it is pre-processed as a job. The Log window-pane (4) displays information about the progress of the processing of the file. It is analogous to the log data of an actual production run and any issues that are listed here will also occur as issues then. The data displayed supplies information which can help you to optimize a workflow by correcting any errors mentioned or paying attention to those processes which took longer.

When processing is complete, the first sheet of the proofed workflow (as it has been designed in Layout) appears in the Sheet Proof Area (3). You can also navigate to the other sheets of the job. This is useful to check for issues within the sheets of the design.

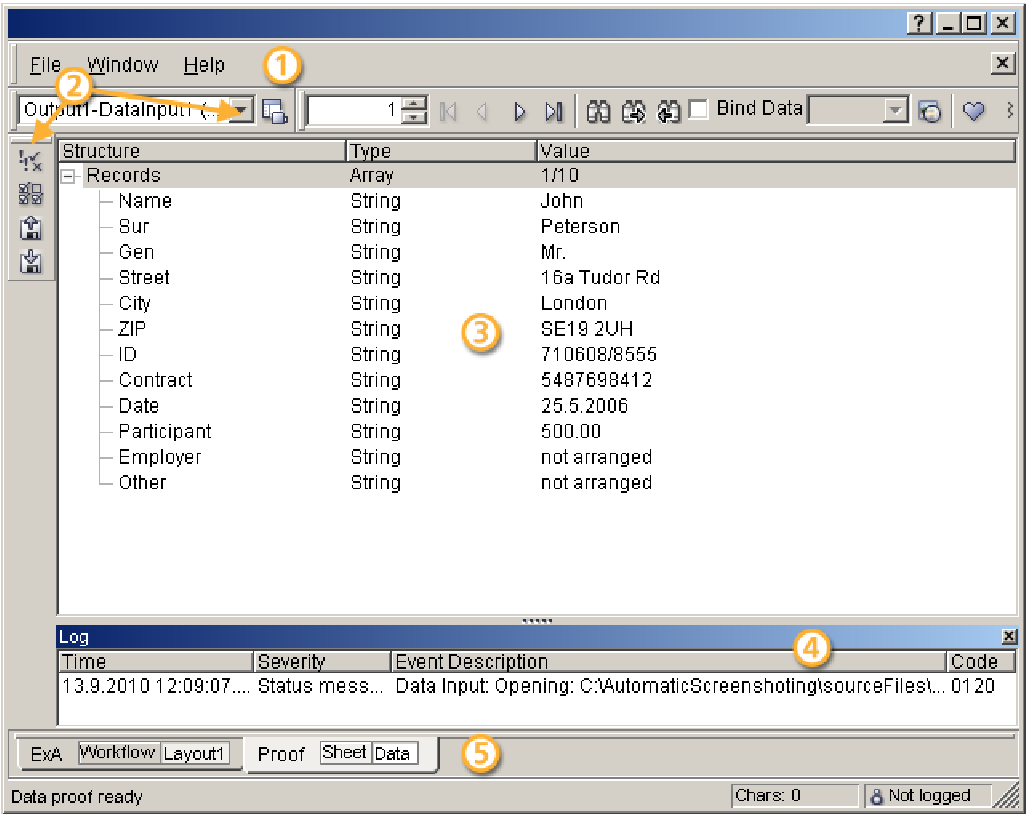
The Sheet Proof menu (1) provides access to general settings; other workflow files to proof; window-panes; different parameters under which to run proof again; and navigation across and through sheets. Many options can be quickly actioned by tools found in the Proof toolbars (2).

The Proof tab (5) shows the open windows within the environment, data-proof is always run alongside proof and can be viewed by navigating to Data.

To learn how to work within the Proof window go to the Working with Inspire Designer chapter.

3.2.2 Data Proof Window

The Data Proof window provides different previews of the data present in the workflow before it reaches the Layout module, whether it has been used in the design of the document or not.



Structure	Type	Value
⊟ Records	Array	1/10
Name	String	John
Sur	String	Peterson
Gen	String	Mr.
Street	String	16a Tudor Rd
City	String	London
ZIP	String	SE19 2UH
ID	String	710608/8555
Contract	String	5487698412
Date	String	25.5.2006
Participant	String	500.00
Employer	String	not arranged
Other	String	not arranged

Time	Severity	Event Description	Code
13.9.2010 12:09:07....	Status mess...	Data Input: Opening: C:\AutomaticScreenshoting\sourceFiles\...	0120

Data proof ready

Chars: 0

Not logged

Figure 3.6 The Data Proof Window – with an Example Workflow already Loaded

Introducing the basic Data Proof window features as they are marked in the figure above:

1. Data Proof Menu
2. Data Proof Toolbars
3. Data Proof Area
4. Log Window-Pane
5. Proof Tab

The Log window-pane (4) displays information about the progress of the data processing performed, no information about Layout or impositioning processing is presented. See the Sheet Proof Window section just above for more details on logging.

When processing is complete, a record of the processed data input is displayed in the Data Proof Area (3). You can navigate to other records and review the data of the whole job.

The Data Proof menu (1) provides access to general settings, other workflow files to proof and environment settings. Some of these options can be quickly actioned by tools found in the Data Proof toolbars (2) alongside tools for navigation through records.

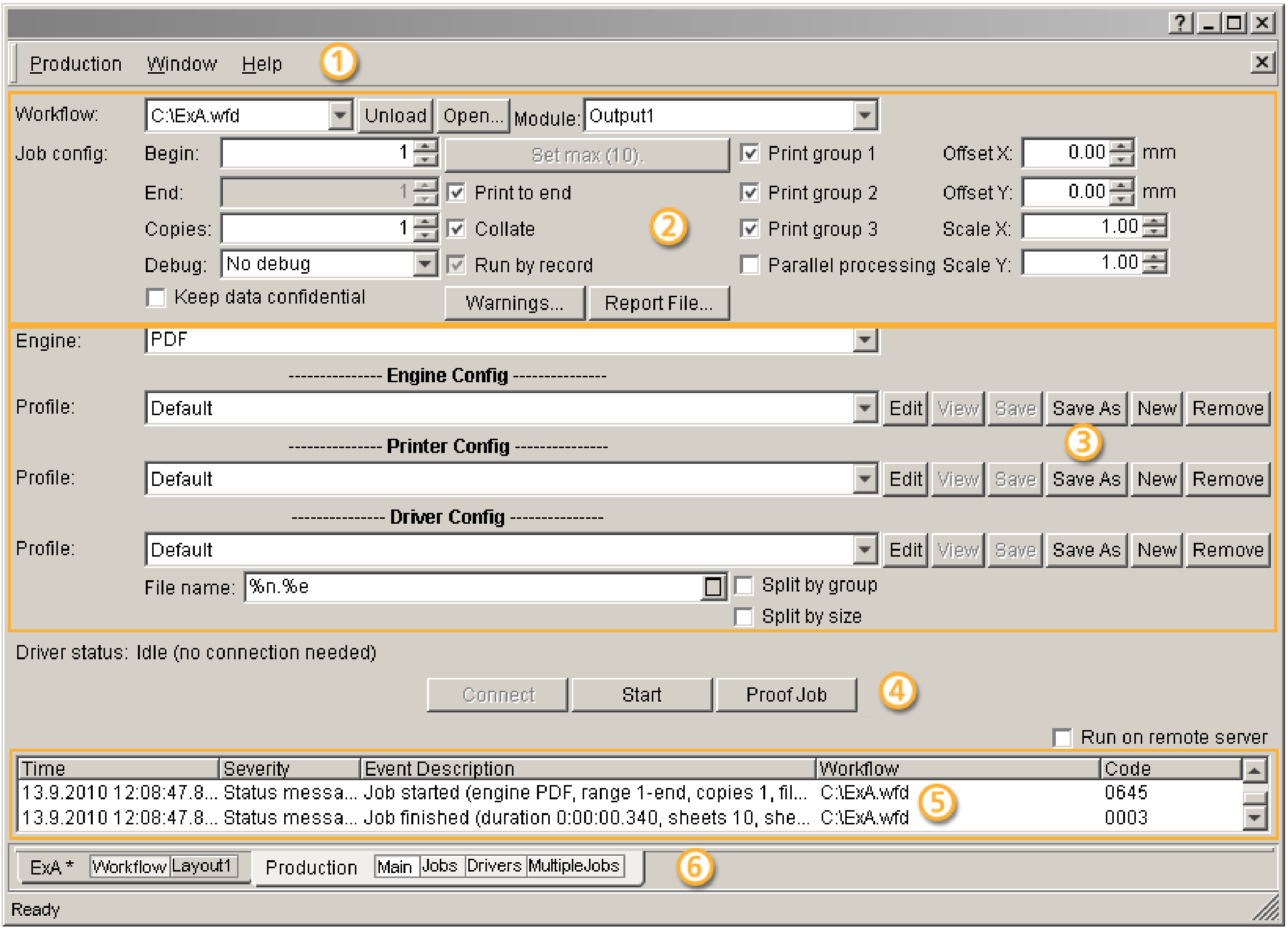
To learn how to work within the Data Proof window go to the Working with Inspire Designer chapter.

3.3 Production

The Production environment is for printing workflows. As Inspire Designer is an application for high-volume printing, the windows of this environment have features that enable the sending of large or multiple jobs to many different types of printers using differing page description languages.

3.3.1 Main Production Window

The Main Production window allows the setting of parameters which relate to the printer with the selection of files and pages to be printed.



Workflow: C:\ExA.wfd Module: Output1

Job config: Begin: 1 End: 1 Copies: 1 Debug: No debug

☐ Keep data confidential

☑ Print to end
☑ Collate
☑ Run by record

☑ Print group 1
☑ Print group 2
☑ Print group 3
☐ Parallel processing

Offset X: 0.00 mm
Offset Y: 0.00 mm
Scale X: 1.00
Scale Y: 1.00

Engine: PDF

-------------- Engine Config --------------

Profile: Default

-------------- Printer Config --------------

Profile: Default

-------------- Driver Config --------------

Profile: Default

File name: %n.%e ☐ Split by group ☐ Split by size

Driver status: Idle (no connection needed)

☐ Run on remote server

Time	Severity	Event Description	Workflow	Code
13.9.2010 12:08:47.8...	Status messa...	Job started (engine PDF, range 1-end, copies 1, fil...	C:\ExA.wfd	0645
13.9.2010 12:08:47.8...	Status messa...	Job finished (duration 0:00:00.340, sheets 10, she...	C:\ExA.wfd	0003

Figure 3.7 The Main Production Window – with an Example Job Already Printed

Introducing the basic features as they are marked in the figure above:

1. Production Menu
2. Workflow Settings
3. Printer Settings
4. Run Buttons
5. Log Area
6. Production Tab

The workflow settings (2) allow the selection of which workflow and pages to print with related parameters. The type of printer engine (i.e. the page description language that the output uses) should be selected from the printer settings (3) and configurations adjusted for the engine, printer driver and, if required, the printer itself.

When a job is then run, by clicking on the **Start** button (4), the progress of the tasks is displayed in the log area (5). Any errors that occur will be highlighted.

The Production menu (1) provides access to general settings, other workflow files to produce and production configurations.

The Production tab (6) shows the other open windows within the environment which can be navigated to. See a brief description of each of these windows in the sections following directly below.

To learn how to work within the Production window go to the Working with Inspire Designer chapter.

3.3.2 Jobs Window

The Jobs window displays the history of the print jobs that have been submitted for production during the current session (i.e. since Inspire Designer was last started) and allows current jobs to be halted.

3.3.3 Drivers Window

The Drivers window displays information about which of the current jobs are using printer drivers, if drivers have been used.

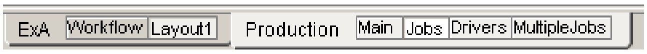
3.3.4 Multiple Jobs Window

The Multiple Jobs window allows the execution of more than one print job at a time. Different workflows can be selected, assigned to differing engines and then run simultaneously.

4 Working with Inspire Designer

The aim of this chapter is to provide a practical guide to the most important methods applied within Inspire Designer. Those that are most commonly required in designing, proofing and producing a workflow.

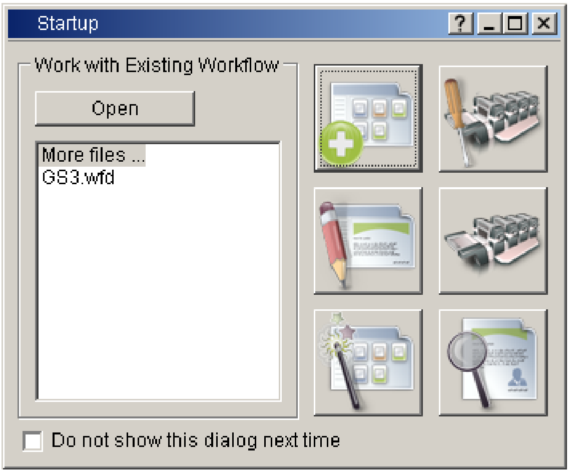
Remember: While you are working with Inspire Designer you can navigate between the open windows using the tabs at the bottom of the window:



4.1 Building a Workflow

The design of a document is defined in a workflow and a WFD file must first be created in order to build the workflow.

To open a new workflow, first start Inspire Designer and, in the Start-up dialog that opens, click on the **New Workflow** button.


Startup
Work with Existing Workflow
Open
More files ...
GS3.wfd
Do not show this dialog next time

Recently saved files are listed for selection in the **Work with existing workflow** field. If **Do not show this dialog next time** is selected, this dialog will no longer appear when Inspire Designer is started and a new workflow will be directly opened instead.

4.1.1 Working in the Workflow Window

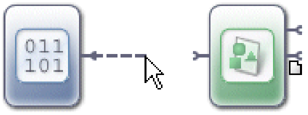
The Workflow window is the main environment for building a WFD file.

To create a WFD file go to the menu **File | Save**, specify a name and location for the file and click on **Save**.

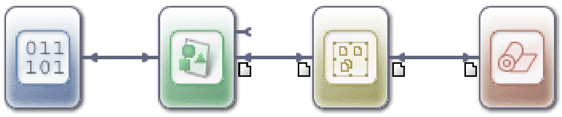
The process of building a workflow starts by adding modules. This is described in the next section.

4.1.1.1 Connecting Modules

1. Each module becomes part of a workflow by dragging and dropping it from the Module Tree into the Workflow Area.
2. Modules should then be connected to other modules in a logical order using the modules' entry and exit points.



The general order of modules across the Workflow Area should be: data input modules first (on the left), then data processing, layout, impositioning and finally output modules (on the right).



Inspire Designer will not allow the connection of modules that do not correspond and so major errors are not possible.

3. Then, the module should be configured according to your needs.

   To open a module and thus reveal its configuration panel(s) you can either double-click on it or right-click on it and select **Edit Module** from the context menu.

   The configuration panel of modules is often tabbed, dividing the properties into categories. There are parameters with values in combo-boxes, edit-boxes, or areas in which new parameters can be added or certain components selected. The parameters are individual to each module and so, when using any module, it is good practice to familiarize yourself with all of the tabs in order to understand the module better.

We will now introduce some methods for working with modules of different functional categories:

4.1.2 Reading Data via Data Input Modules

The first modules in a workflow are, most usually, data input modules. Because Inspire Designer is foremostly a variable-data printing application, the function of these modules is to identify a source for that data so that it can be read into the workflow.

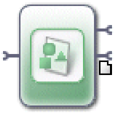
There are many different data input modules, each one geared to read different formats of data files. Therefore, each has a different configuration. A selection of the available data input modules is employed in the Examples chapter of this guide.

4.1.3 Manipulating Data with Data Processing Modules

Once the data has been read into the workflow it will not necessarily be in the correct format for the intended document's design. Data processing modules can be placed after the data inputs in order to manipulate that data so that it will act exactly how it is needed to.

There are many different data processing modules, each one geared to perform a different type of manipulation. For example, some merge data sources, some filter data and some can change the *data-type*. Therefore, each module has a different configuration. A selection of the available data processing modules is employed in the Examples chapter of this guide.

4.1.4 Designing Pages in the Layout Window



The Layout window is used to design *pages*. It allows the addition of text, images and other objects to different page designs which are structured using the Layout Area and the Layout Tree.

4.1.4.1 Working in the Layout Window

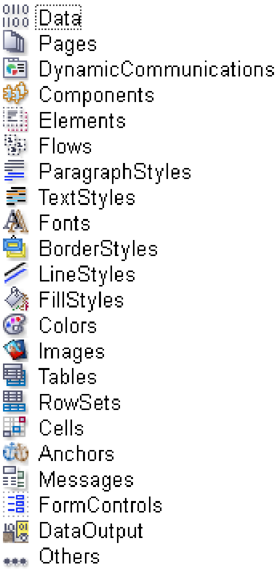
To open the Layout window from the Workflow window, double-click on the Layout module used in the Workflow Area or go to the menu **Workflow | Open Layout**.

To close the Layout window and return to the Workflow window, go to the menu **File | Close Window**.

It may happen that the window-panes within the Layout window environment get re-arranged. In order to return to the default arrangement, go to the menu **Window | Reset Environment**.

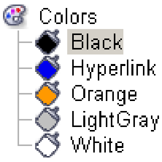
4.1.4.2 Navigating the Layout Tree

As well as managing the visual appearance of a design in the Layout Area, it is important to understand the relationship between the objects of the design. This can be done using the Layout Tree. As each new object is inserted into the Layout a corresponding node is added to the Layout Tree. Many objects are grouped in families:



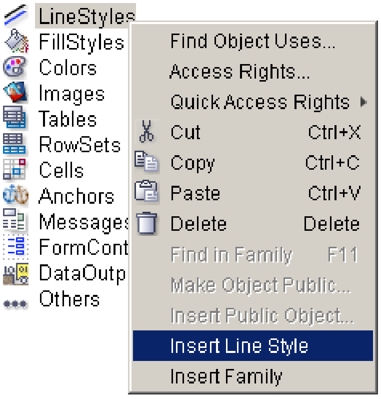
Data
Pages
DynamicCommunications
Components
Elements
Flows
ParagraphStyles
TextStyles
Fonts
BorderStyles
LineStyles
FillStyles
Colors
Images
Tables
RowSets
Cells
Anchors
Messages
FormControls
DataOutput
Others

By clicking on ⊞ to expand the branches, you can explore which objects of each given family have been created:

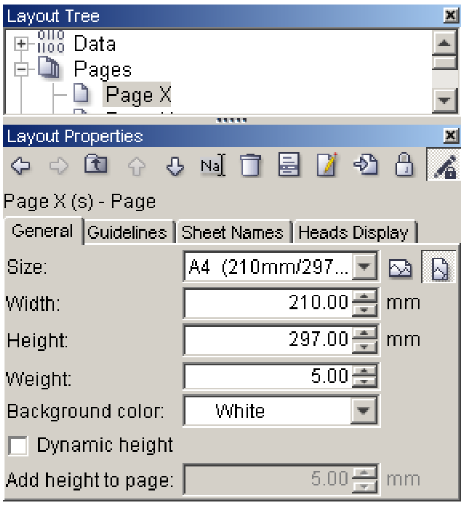


Colors
Black
Hyperlink
Orange
LightGray
White

By right-clicking on an object, a context menu is revealed:



By selecting an object, its properties are displayed in the Layout Properties panel:



Page X (s) - Page

General | Guidelines | Sheet Names | Heads Display

Size: A4 (210mm/297...

Width: 210.00 mm

Height: 297.00 mm

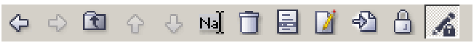
Weight: 5.00

Background color: White

☐ Dynamic height

Add height to page: 5.00 mm

The Layout Properties toolbar at the top of the properties panel aids the navigation between objects as well as the editing of them:



4.1.4.3 Defining Pages

The process of designing a layout begins with defining the design space, i.e. the different pages of the design.

By default, when a workflow is printed it will typically repeat all of the pages of the design by the records that are being read into it, i.e. the number of repetitions is equal to the number of records. This means that only one set of pages needs to be designed and a page order set-up between them (e.g. first to second to third page). Consequently, when each new record is read from the data input, the page order automatically starts again from the beginning.

To understand the idea of records you can imagine a simple example: creating a letter (one page) for a certain number of clients (e.g. 100). Your data file would contain 100 names and address details for these clients. Each of these 100 entries is one record. You would design just one layout page and it will be repeated 100 times, each time for a different client, when printing/proofing the output.

When the Layout window opens for the first time, a new page is displayed in the Layout Area and this page can be seen in the Layout Tree below the Pages family (by default named "Page1").

Selecting this page in the Layout Tree reveals its properties panel where you can set the **Width** and **Height** of the page which is by default 210 x 297 (A4 paper size). These properties can be adjusted to match the size of the intended printing medium.

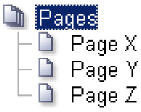
Setting Page Order

When there is more than one page in a Layout, a page order can be set up so that those pages will be printed in a defined order, which can be a different order than the simple one-by-one sequence. One advantage of this is that any page design can be selected as the next page in a series, including the ability to repeat certain pages if required.

To determine the page order, select the Pages family on the Layout Tree and go to the General tab on its properties panel.

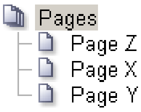
There, by default, *Simple* is set in the **Page order** combo-box. This determines that:

1. The set of pages will be repeated for each record.
2. The pages will follow each other as they are ordered in the Layout Tree. E.g.:
   - Page order X, Y then Z:



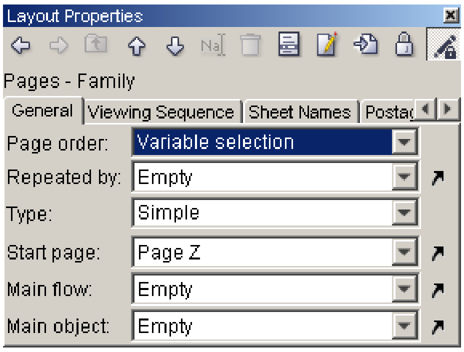
Pages
- Page X
- Page Y
- Page Z

   - Page order Z, X then Y:



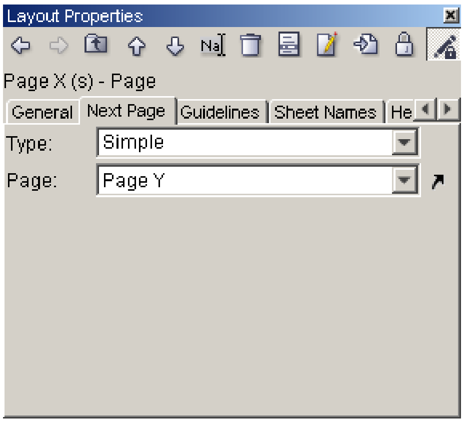
Pages
- Page Z
- Page X
- Page Y

An alternative setting in the **Page order** combo-box is *Variable selection* which, when selected, allows the further setting of page order parameters:



Page order: Variable selection
Repeated by: Empty
Type: Simple
Start page: Page Z
Main flow: Empty
Main object: Empty

Here, the **Start Page** should be selected from the combo-box menu, in order to define the first page of the sequence. The sequence is then further defined in the properties of each individual page, via the **Page** combo-box on the Next Page tab on that page's properties panel:



Page X (s) - Page

Type: Simple

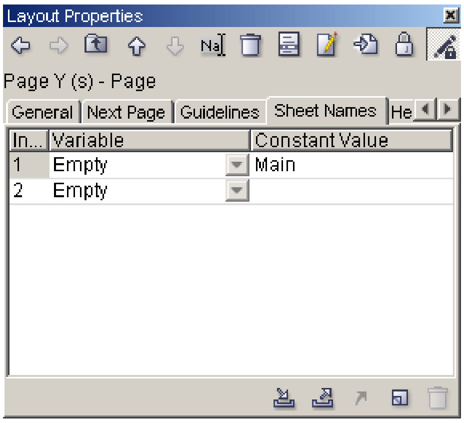
Page: Page Y

It is also possible to redefine how the series of pages will be repeated, as opposed to a repetition for every record. This is done by the **Repeated by** combo-box of the Pages family, which allows the selection of another *variable* of array *data-type* to repeat the set of pages by. This is a more advanced feature that will be employed in Example B of this guide.

Assigning Sheet Names

Sheet names are identifiers for the different pages of a design. They are passed along the workflow as part of the definition of each sheet. They can then be used by other modules in order to identify which sheets to manipulate or to carry other values that might need to be used in connection with a particular page design.

Sheet names are given to individual pages of a design on the Sheet Names tab of the page's properties panel:



Page Y (s) - Page

In...	Variable	Constant Value
1	Empty	Main
2	Empty	

The first sheet name for a page should be entered in the first row and it should relate to the intended media that the document will be printed on. Therefore, if all sheets will be printed on the same media (e.g. A4 paper with portrait orientation) then all pages should have the same sheet name 1 entered. But, if each page will be printed on a different medium (e.g. three pages: one A4 portrait, one A4 landscape and one envelope) then each page should be given a different sheet name 1. Sheet name 1 can then be used in the Output module to define the media selection. All pages are given the same sheet name 1 (*Main*) by default.

It is possible to apply more that one sheet name in order to identify pages for multiple purposes. For example, all pages have the same size, but one page always needs to be printed twice so that it can be filed. Such a repetition of a single page can be defined using an impositioning module, but that page must have a unique identifier (sheet name) in order to be able to define this in an impositioning module.

Sheet name 1 and sheet name 2 (i.e. the sheet names with index number *1* and *2*) are reserved for the individual pages of the design. Sheet names with higher indexes (e.g. *3, 4, 5*) can also be applied to all of the pages as a group via the Pages family on the Layout Tree.

Sheet names applied to the whole Pages family can be used to identify each repetition of the full set of pages in the design, i.e. groups of sheets bound by a common value. For example, a sheet name with the index *3* can be used to identify which records from a data file have been printed using those pages, by assigning the sheet name a variable that will return a unique value for each record.

Sheet names are employed in Example A and Example B.

4.1.4.4 Editing Pages

Objects that are placed on pages are listed in two places in the Layout Tree:

- Under their family node.
- Under the node of the page that they have been placed on.

  This provides an overview of the complete content of a page from the Layout Tree. Furthermore, the order of the objects at this location also has an influence on processing.

For example, if you have an image which is placed on two pages, it will be listed under the according nodes of those two pages and you can also find it under the Images family node. Objects from the family nodes can be re-used on multiple pages. It is recommended not to delete objects from the family nodes unless you are sure that they are not being used on any of the pages of the document that you are editing and that you will not need them at a later stage of design.

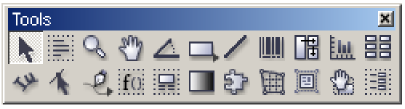
When creating a document of many different pages, the tree structure of Pages can grow to a very large size. Therefore, it becomes increasingly important to rename the objects in order to aid recognition and navigation through the pages. See the Naming Objects section for more details.

In the simplest of terms, there are three kinds of objects:

1. Objects that define the design space (e.g. a page).
2. Objects that define areas (positions and paths) within the design space (e.g. a shape drawn on one of the pages).
3. Objects that define content that can be used within objects that define areas (e.g. flows, images and line styles).

Arranging Objects

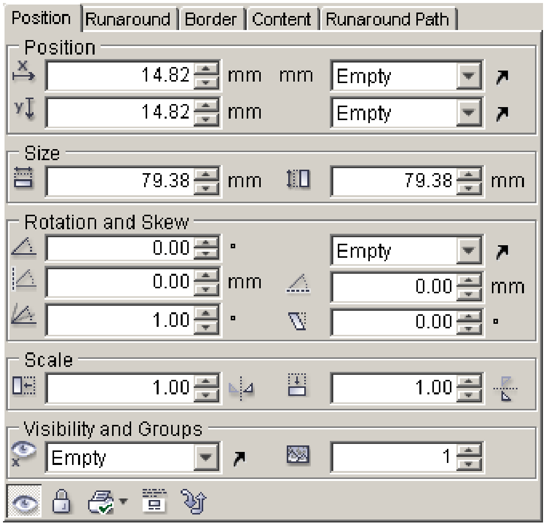
Objects are placed and arranged in the Layout Area with the aid of tools. Various tools can be found in the Tools toolbar which is located, by default, down the left-hand side of the Layout window:



Different objects require different tools and many of these will be introduced in context, as we continue through this guide.

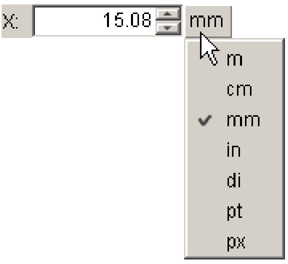
The default tool is the **Selection Tool** and this tool must be active in order to move or resize any object that has been placed in the Layout Area. Click on it to activate it.

Objects that are placed on pages (e.g. flows, images etc.) are placed into area objects (e.g. flow areas, image areas etc.) which determine their position, shape and size on that page. Each of these area objects has a Position tab on its properties panel:



This Position tab can be used to alter the position, size, rotation and further attributes accurately by entering new values in the respective edit-boxes.

Throughout the attributes of Layout, measurement units can be changed by clicking on the unit symbol:

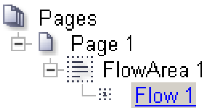


Adding Text

Passages of text are objects called *flows* which are placed on pages using *flow areas*. Flow areas are drawn using the **Flow Area Tool**. To do so:

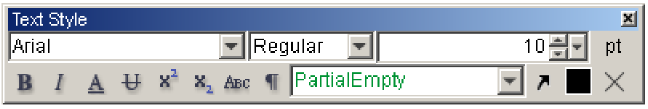
- Select the **Flow Area Tool**.
- In the Layout Area, click and drag the mouse to draw a flow area.
- Type the text at the cursor point to enter text into the flow.

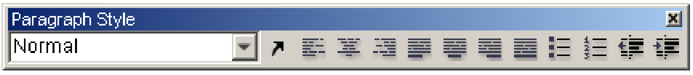
Remember: All flows that are added to the page are grouped below that page's branch in the Layout Tree:



In the screenshot of the Layout Tree above, notice that the flow area contains the flow on the page. This is because the flow is kept as a separate object from the flow area. Such an arrangement allows a flow area to easily replace one text passage with another if needed. It also allows many text passages to be kept in the layout without being used on any page, so that they can be used if required.

The text within flows can be formatted and have styles applied using the Text Format and ParaStyle toolbars found at the top of the Layout window:





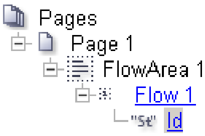
Inserting Variables

Variables are key to variable-data printing. They contain data values that change repeatedly as a workflow is processed. Typically, the value of a variable will change for each record.

If a data file is used in the workflow, variables appear as objects in the Layout Tree under the Data family. They return their value as text characters if they are placed into a text passage, i.e. into a *flow*. To do this:

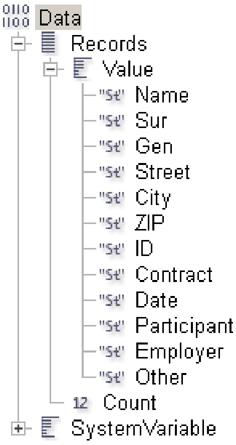
- Select the **Flow Area Tool** .
- Select the variable from the Layout Tree.
- Drag it to a position in a flow area, within the flow.

Remember: All variables that are added to the page are grouped below that page's branch in the Layout Tree:



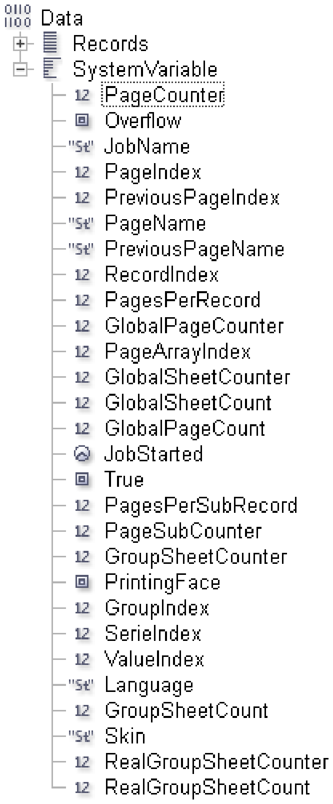
There are different categories of variables that can be used in the design of the pages of a document:

1. Record Variables – Are taken from any data files that are read into the workflow. They return a new value with each record that is read. They can be found in the Layout Tree where they have been named according to the data input definition:



Record variables are employed in Example A and Example B.

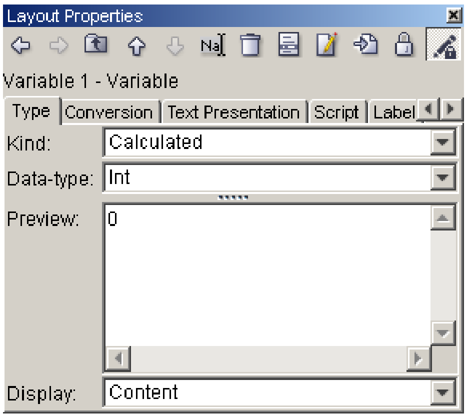
2. System Variables – Return values that relate to the document itself. These are always available for use in Layout:



Data
- Records
- SystemVariable
  - PageCounter
  - Overflow
  - JobName
  - PageIndex
  - PreviousPageIndex
  - PageName
  - PreviousPageName
  - RecordIndex
  - PagesPerRecord
  - GlobalPageCounter
  - PageArrayIndex
  - GlobalSheetCounter
  - GlobalSheetCount
  - GlobalPageCount
  - JobStarted
  - True
  - PagesPerSubRecord
  - PageSubCounter
  - GroupSheetCounter
  - PrintingFace
  - GroupIndex
  - SerieIndex
  - ValueIndex
  - Language
  - GroupSheetCount
  - Skin
  - RealGroupSheetCounter
  - RealGroupSheetCount

System variables are employed in Example B.

3. User-Defined Variables – Can be created as required. To do this, right-click on any variable, on the level of the Layout Tree at which it is required, and select **Insert Variable** from the context menu. Then define the new variable in its properties panel:



Layout Properties

Variable 1 - Variable

Type | Conversion | Text Presentation | Script | Label

Kind: Calculated

Data-type: Int

Preview: 0

Display: Content

The most frequently used kinds of user-defined variable are:

- *Calculated* – A script-defined variable, usually containing a brief calculation expression. The calculated value can then be used like any other variable.

  If the calculation uses values returned by other variables, the calculated variable must be created in a dependency level below those variables in the Layout Tree. This is necessary to ensure that the calculated variable will be processed after the other variables used in the calculation, i.e. the correct values will be used in the calculation.

- *Global* – A variable that is linked to another variable and used to remember the value of that variable until it is processed again. At this point the new value that that variable returns will be added to the previous value, creating an accumulative sum.

  Variables are processed according to their position in the Layout Tree (i.e. their position under the node of the page on which they are placed). Thus, the position in the design can play an important role. The advanced functionality of global variables, where the placement matters, will be shown in Example B.

User-defined variables are employed in Example B.

Inserting Images

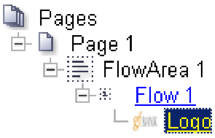
Images are external files that are inserted into the Layout Tree to become objects that can be used in the design. To insert an image go to the menu **Insert | Insert Image** and select an image from the Open dialog.

Images are placed on pages using *image areas*. Image areas are created automatically when an image is dragged to a page. To do so:

- Select the image from the Layout Tree from within the Images family with the **Selection Tool** active.
- Drag it to a position on a page.
- Drag the boundaries of the image area to adjust its size.

Images can also be added directly to flows and selected as fill styles. However, the size of the image's appearance in the design can only be adjusted by an image area, which is not used when employing them in flows or fill styles. In such cases, the actual size of an image has to be managed in another software application before it is inserted.

Remember: All images that are added to the page are grouped below that page's branch in the Layout Tree:


Pages
Page 1
FlowArea 1
Flow 1
Logo

Inserting images will be employed in Example A and Example B of this guide.

Structuring with Tables

A *table* is a collection of *flows*. Each flow is placed inside an individual cell. *Cells* are organized into row sets, which are groups of rows with similar properties (e.g. header rows), and these *row sets* are collected together by a greater row set that governs the behavior of all of the row sets together. Finally that greater row set is held by a table which governs properties that relate to the table as a whole.

Tables, row sets and cells are all independent objects and have different sets of properties that, combined, control the appearance of the table.

Table creation involves:

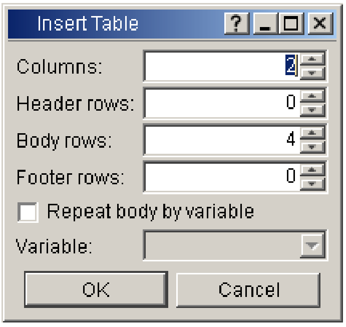
- Inserting Tables
- Formatting Tables

Inserting Tables

Tables are placed inside flows, which are placed on pages using *flow areas*. When a flow area is selected then a table can be inserted as part of the flow within it. To do so:

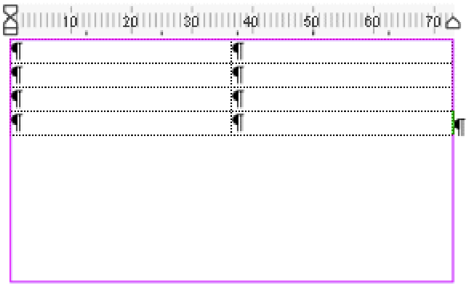
- Select the **Flow Area Tool** .
- Click into an already created flow area where the table should be placed.
- Go to menu **Table | Insert Table**.

The Insert Table dialog opens:



Columns: 2
Header rows: 0
Body rows: 4
Footer rows: 0
□ Repeat body by variable
Variable:

Body rows are the rows that make up the bulk of the table. When adding a table that does not require a header or footer, simply enter the number of required **Columns** and **Body rows** and click on **OK**. The table will appear inside the flow:



It can be seen from the paragraph markers (¶) in the above screenshot that there are nine defined paragraphs in the flow area, i.e. nine flows in the flow area. The eight flows of the table are nested inside one first-level flow.

Paragraph marks are viewable by selecting the **Show Hidden Characters** ¶ icon from the Text Format toolbar.

In the Insert Table dialog it is possible to define header and footer rows. This can be useful to do in the following situation: If you have a table overlapping to a second page/flow area or more, you might want to distinguish between the First Header (only displayed on the first page/flow area where the table appears) and the Header (displayed on all other pages/flow areas) and between the Last Footer (displayed on the last page/flow area where the table appears) and the Footer (displayed on all other pages/flow areas). This behavior is illustrated in the screenshot below showing how the different header and footer rows would be used in a table spread across three pages/flow areas:

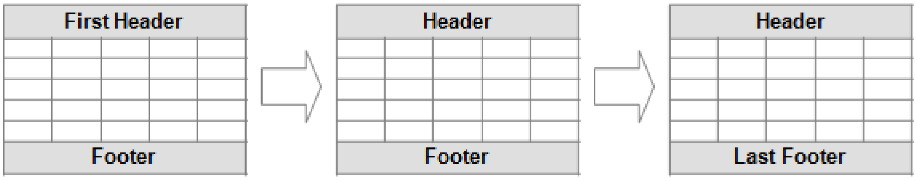
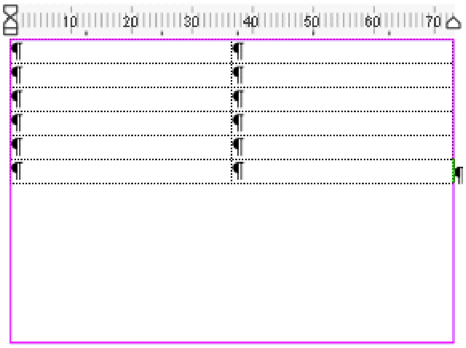


Figure 4.1 The Header/Footer row set and its use of Headers and Footers

To demonstrate this behavior, we will create a table with one header and one footer row in addition to four body rows (six rows altogether). Specify the following values in the Insert Table dialog:

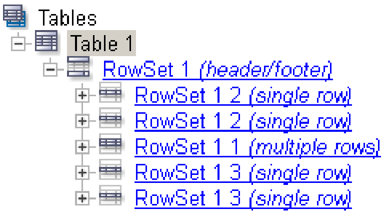
- **Columns** – *2*
- **Header rows** – *1*
- **Body rows** – *4*
- **Footer rows** – *1*

By default the second set of header and footer rows cannot be seen in the Layout Area (to have both header and both footer rows displayed in the Layout Area, select the *header/footer* row set in the Layout Tree and on its properties panel select the **Display all rows** check-box):



Notice that in the table above, there are the six rows that we defined in the Insert Table dialog.

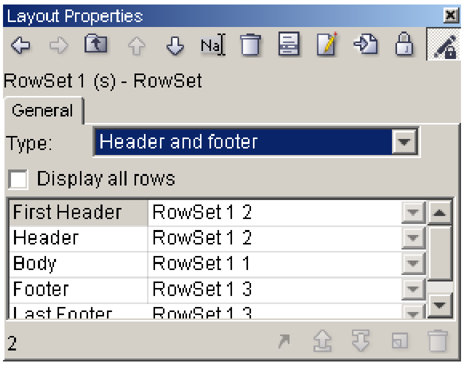
Remember: When a table appears on a page, the properties of that appearance (e.g. the borders) are actually properties of the individual cells. Also, the structure in the Layout Tree appears to be different than what is seen on the page (see the explanation below the screenshot for details):



Tables
Table 1
RowSet 1 (header/footer)
RowSet 1 2 (single row)
RowSet 1 2 (single row)
RowSet 1 1 (multiple rows)
RowSet 1 3 (single row)
RowSet 1 3 (single row)

Notice how, in the Layout Tree, the table is made-up of one large row set containing smaller row sets. The properties of row sets control the logic of the cells in a table.

In our example table, there is now one header row set (represented by two row sets of the same name), one body row set and one footer row set (represented by two row sets of the same name) that combine to make up the *header/footer* row set. In this *header/footer* row set, the header is duplicated – one for the First Header of the table and one for the Header that will appear at the top of the table if it overflows. Similar for the footer – one for the Last Footer and another Footer in case the table overflows. The body row set can be expanded to see the 4 body rows which are present in this example and visible in the screenshot of the table itself. This arrangement is better evident on the properties panel of the header/footer row set:



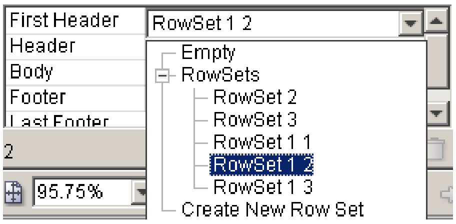
Type: Header and footer

□ Display all rows

First Header	RowSet 1 2
Header	RowSet 1 2
Body	RowSet 1 1
Footer	RowSet 1 3
Last Footer	RowSet 1 3

2

This structure of tables makes it easier to replace row sets with others, that can be selected or created via the drop-down menu that appears in each row:



First Header	RowSet 1 2
Header	
Body	
Footer	
Last Footer	

For example, it might be needed to replace the **Header** row set (for overflowing tables) with one of a different design to show some different information such as the text "Continued from previous page".

Checking the **Display all rows** check-box shows all of the rows at the same time within the table in the Layout Area, to allow editing of all rows there.

The reason for this structure of Header/Footer tables is that body rows of a table can be repeated an unknown amount of times (depending on the data being used), making it difficult to predict if or when the table might overflow. Also, tables should be able to handle differing amounts of body rows. In this case, it is not possible to add header and footer rows in the right place manually.

If it is necessary to display varying amounts of data in a table, it is not possible to create a fixed amount of body rows for it. But, it is possible to determine that body rows should keep repeating as often as necessary to display all of the data. This can be done by checking **Repeat body by variable** in the Insert Table panel and selecting an array (containing the data) from the **Variable** combo-box menu just below.

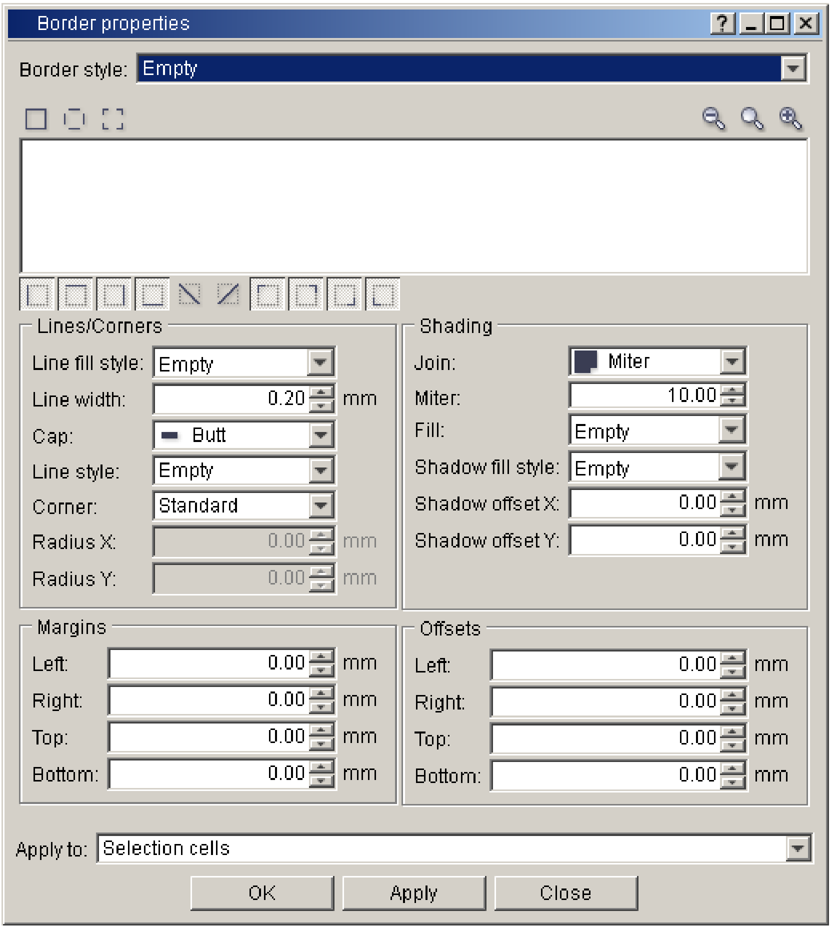
Inserting tables will be employed in Example B of this guide.

Formatting Tables

The visual appearance of the table (i.e. borders and content) is controlled by the cells. Once a table has been inserted, it is common that it should also be formatted, i.e. fill and border styles added to its cells. When cells are selected, they can be formatted. To do so:

- Select the **Flow Area Tool** .
- Select the cells which should be formatted.
- Go to the menu **Table | Table Border Style**.

The Border Properties dialog opens:



Border style: Empty

Lines/Corners

- Line fill style: Empty
- Line width: 0.20 mm
- Cap: Butt
- Line style: Empty
- Corner: Standard
- Radius X: 0.00 mm
- Radius Y: 0.00 mm

Shading

- Join: Miter
- Miter: 10.00
- Fill: Empty
- Shadow fill style: Empty
- Shadow offset X: 0.00 mm
- Shadow offset Y: 0.00 mm

Margins

- Left: 0.00 mm
- Right: 0.00 mm
- Top: 0.00 mm
- Bottom: 0.00 mm

Offsets

- Left: 0.00 mm
- Right: 0.00 mm
- Top: 0.00 mm
- Bottom: 0.00 mm

Apply to: Selection cells

The Border Properties dialog is for selecting existing border styles or creating new ones. It contains the formatting attributes, such as line styles and fill styles, which, when applied, are combined together to create a single border style. That border style can subsequently be selected for other tables and objects.

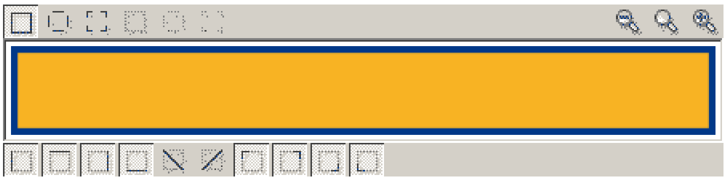
The selection of already existing border styles can be done via the **Border style** combo-box. When creating a new style or adjusting an existing one, also note the changes that occur in this combo-box:

1. When you first open the Table Properties dialog for a new table it will read *Empty* meaning that there is no style applied.

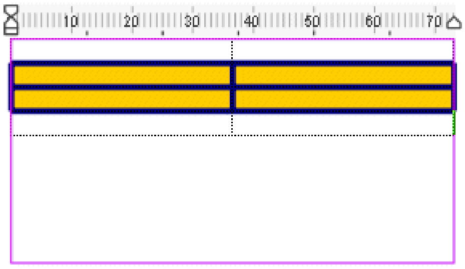
2. When you change any of the properties, it will read *Modified* which means that the current selected style has been modified by you but has not yet been saved.

3. Finally, when you click on **OK**, the border style becomes an object that can be found in the Layout Tree under BorderStyles family node.

A border style can be adopted as a property for individual cells or the table as a whole. The preview area at the top of the Border Properties dialog displays an example of a single cell, as the format parameters are altered:



View in the Border Properties dialog



Resulting formatting of selected cells

It is important to note that the styles will be applied to the selected cells individually. Each cell has four borders, i.e. in between two adjacent cells there are two borders which will overlap, which is of importance when two different styles are used for adjacent cells.

The Cell Properties dialog (go to the menu **Table | Table Cell Properties**) contains extra cell properties that are not components of border styles and can also be found on the properties panel of individual cells. The advantage of applying these properties here, however, is that many cells can have their properties adjusted simultaneously.

The **Apply to** combo-box available at the bottom of both of these dialogs allows the selection of which cells the newly adjusted parameters should be applied to. Some frequently used options are:

- *Selection cells* – Parameters are applied to all cells that have been selected before the dialog was opened.
- *Table cells* – Parameters are applied to all of the individual cells within a table.
- *Table* – (For border styles only) Parameters are applied to all of the table, i.e. the line style will be applied to the outer boundary of the table and the fill style will be applied to all cells.

  However, if *Table* is used and a cell already has its own style applied then that cell's style is kept.

> NOTE Because border styles are objects that employ both line styles and fill styles then, when creating a new border style, new line styles and fill styles are created at the same time. This is important to remember if you want to rename objects in the Layout Tree.

Table formatting will be employed in Example B of this guide.

Drawing Shapes

Among other objects, basic geometric shapes can be added as a part of the document design. Fill styles and line styles can be used to customize their appearance. Rectangles, triangles, ellipses, stars or polygons can be drawn using the **Shape Tool** . To do so:

- Right-click on the **Shape Tool** and select the required shape.
- Draw the shape by holding down the mouse button and dragging the pointer across the Layout Area.
- Go to the shape's properties panel and select (or create new) fill and line styles.

The Polygon and Star shapes also prompt you to enter how many sides will form them.

Shapes will be employed in Example B of this guide.

Creating Styles

Styles are content defining objects that are used by other objects to enhance their appearance. They include:

- Paragraph Styles
- Text Styles
- Fill Styles
- Border Styles
- Line Styles

Often they are created when setting the properties of objects to which they are to be applied. Whenever a new style is applied a new style is created as an object under its family node in the Layout Tree. That style object can then be quickly selected for other objects when it is required that they have the same style.

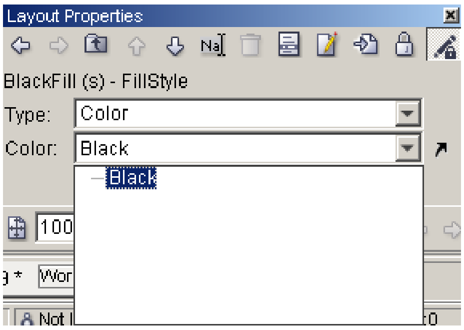
It is recommended to manage the styles with respect to keeping the amount of styles used to a minimum. I.e. instead of creating a new style every time that you require to fill an object, consider selecting a style that was created previously and, upon completion of a design, remove all styles that have been created but are no longer used.

Styles can also be prepared in advance by creating them via the context menus in the Layout Tree. Right-click on the appropriate family (e.g. TextStyles) and select the according Insert command (e.g. **Insert Text Style**) from the context menu.

Styles will be employed in Example B of this guide.

Managing Colors

Colors are content defining objects that are used by fill styles. Each time a new color is used in the design of a document a new color object is created under the Colors family node in the Layout Tree. That color can then be selected in the properties panel of the fill style:



Layout Properties

BlackFill (s) - FillStyle

Type: Color

Color: Black

It is recommended to manage them with respect to keeping the amount of existing colors to a minimum. I.e. instead of creating a new color every time that you require to fill an object, consider selecting a color that was created previously. Upon completion of a design, remove all colors that have been created but are no longer used. Such practice is especially beneficial when making considerations about the colors that will actually be used by a printing device. For example, only a selection of colored inks will be used and the design should use only those colors.

For colors it is even more important than for styles that they are prepared beforehand. The colors that will be used in the design can be added via the context menu in the Layout Tree. Right-click on the Colors family and select **Insert Color** to do so. Then the actual colors can be defined via their properties panel.

Clicking on the **Change** button located there opens the Color dialog which allows either the setting of new RGB, CMYK, HSB or CIELab values; or the assignation of spot colors, dithered spot colors or DeviceN colorants.

Colors will be employed in Example B of this guide.

4.1.4.5 Overflowing Page Content

In addition to defining the page order, it is also important to define the manner in which the content on the pages will flow from one page to another.

Usually, when talking about overflowing the content of pages, this means flows because the most likely content to overflow is passages of text. Therefore, defining overflows is usually defining that: "if flow area X is filled then the remainder of the flow should be shown in flow area Y". A series of overflows can involve many flow areas, which can be on the same page or between pages.

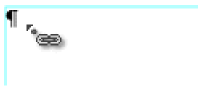
Remember [29]: The repeated body of a table is a special case where the content of the flow is the repeated table, not just simple text.

Overflows are defined using the **Flow Order Tool** .

To define an overflow:

1. Select the **Flow Order Tool** and click once on the flow area that contains text which does not fully fit into it.
2. Navigate to the next flow area, i.e. to the flow area where the content of the flow should go next, and hover the mouse pointer over it.

   A flow order mouse pointer should now be shown over that flow area:

3. Click once to complete the definition.

   If a flow order mouse pointer is not showing, then the previous flow has not been selected properly.

You can check to see that the correct overflowing behavior has been assigned on the Content tab of the flow areas' properties panel. The **Content** combo-box of all of the flow areas in the series should have exactly the same flow selected.

Depending upon whether an overflow is between flows on the same page or between flows on separate pages, different color arrows indicate that a flow order has been set. These arrows are visible when the **Flow Order Tool** is active:

- Blue arrows – Overflow direction on the same page.
- Pink arrows – Overflow to/from another page.

Overflowing content will be employed in Example B of this guide.

4.1.4.6 Naming Objects

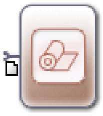
As objects are created or inserted, they are given a default name. This default name is in the format of the name of the object type plus a serial number, e.g. if it is the third shape object to be created it will be called Shape3.

As the design progresses the number of objects that are part of it grows. Often it is required to navigate through objects or select objects to make up others, and so it becomes almost essential to rename the objects.

The best way to rename objects is renaming them in the Layout Tree as they are created. Any new objects can easily be found there (especially if you have renamed the previously created objects). Selecting an object and clicking on the **Rename Object** icon from the Layout Tree toolbar, allows the editing of the name. Alternatively, <**F2**> is the keypad shortcut for name editing.

To rename workflow modules, right-click on them and select **Rename Module** from the context menu. This opens the Rename module dialog where you can enter a new name for the module.

4.1.5 Configuring the Output Module



Double-clicking on the Output module in the Workflow Area opens it.

In the output module, it is possible to define the *media* that will be printed on and then define some parameters for the *device* that it will be printed by. The following two sub-sections describe how to properly set up media and a device in the Output module.

In addition to that, you can pre-define already in the Output module what profile will be used for proofing your workflow and what engine will be selected when producing a document. This is very helpful if more Output modules are used within one workflow.

4.1.5.1 Media

Media definition in the Output module consists of setting workflow parameters that aid the selection of the correct media for the pages of the design to be printed on. Therefore, first consider the printing media. For example, pages designed in landscape size would probably be printed on paper in a landscape orientation.

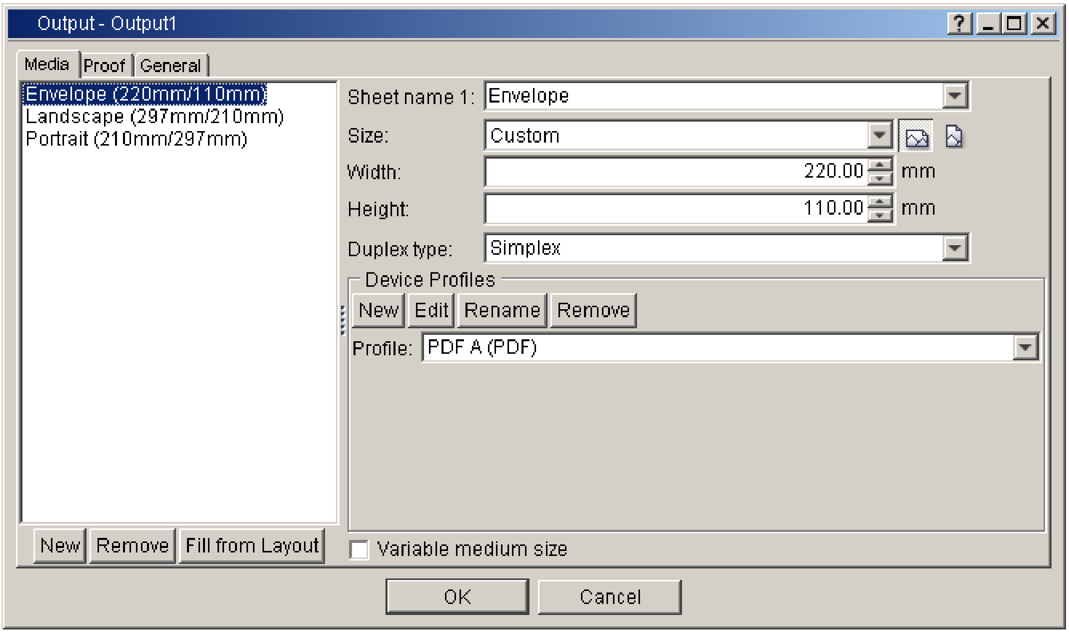
If the workflow's design consists only of pages which all have the same size and will all be printed on the same media, then there is no need for defining media pages here in the Output module. However, if you either have designed pages of different sizes; want to print different pages of the design to differing media; or want to define some Device Settings, then one media page must be created for each media that the pages will be printed on.

To create media pages, follow these steps:

1. In the Layout module, assign a sheet name 1 to each of the pages of the design. All pages that will be printed on the same media should be given the same sheet name 1.

2. Return to the Output module and click on **Fill From Layout**. For each different sheet name 1 in Layout a new media page will be automatically created and the according fields (**Sheet name 1**, **Size**, **Width** and **Height**) correctly filled according to the values already set in Layout.

For example, if there were three different page sizes designed in layout (Portrait, Landscape and Envelope) you could set corresponding values for **Sheet name 1** for each of them (*Portrait*, *Landscape* and *Envelope*). Then, in the Output module, three corresponding media pages named *Portrait*, *Landscape* and *Envelope* and using the sizes defined in Layout would be created after clicking on **Fill from Layout**:



Output - Output1

Envelope (220mm/110mm)
Landscape (297mm/210mm)
Portrait (210mm/297mm)

Sheet name 1: Envelope
Size: Custom
Width: 220.00 mm
Height: 110.00 mm
Duplex type: Simplex
Device Profiles
Profile: PDF A (PDF)
□ Variable medium size

Media pages will be defined in Example A and Example B of this guide.

4.1.5.2 Device

Device definition in the Output module consists of passing some additional parameters to the printing device in order to define the handling of media pages. These parameters can determine how the pages should be printed or some extra operations that should be performed. It follows that each printer type has a different set of parameters that can be set. The device settings are adjusted only if it is required for the desired output. In some cases the settings of the printing engine properties in Production is enough.

Because of the many types of page description languages that Inspire Designer can communicate in, we will not introduce the parameters for all of the different printer types here. However, standard practice for creating a device profile is:

1. Create at least one media page. See the Media section above.

2. Create a device profile by clicking on the **New** button in the **Device Profiles** area.

3. Configure the profile by adjusting the printer specific parameters that appear on the Output dialog. This profile will then be saved in the Output module for selection whenever the current workflow is being edited.

4. Select the profile in the **Profile** combo-box.

One device profile can be created for each type of printer. This will automatically be selected when the workflow is produced using the corresponding engine in Production.

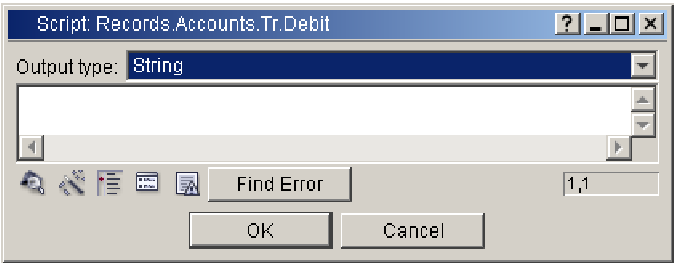
Device profiles will be defined in Example A and Example B of this guide.

4.1.6 Scripting Extra Parameters

Scripting in Inspire Designer is based on C++ and some knowledge of this language is necessary to be able to produce your own scripts. However, the need for scripting in Inspire Designer is kept to a minimum because most of the functionality, required to design workflows, exists in the GUI. When scripts are needed they are usually only basic. These are created in context and applied to a specific object or component of the workflow.

Scripting increases the possibilities of workflow design in all-manner of ways. The script dialogs or tabs, that can be called on in various modules, allow the simple addition of expressions and fragments of script that will be automatically processed in the correct context. This can allow anything from a simple command to return a Boolean value to executing a separate application to process large amounts of data.

A typical script dialog or tab has some helpful script composition aids:



The **Find** icon opens the Find & Replace dialog to correct repetitive errors. The **Wizard** icon opens the script wizard to help produce basic scripts and the **Find Error** button checks any script that is written for syntax errors. Debugging functionality is available in Proof.

Some simple scripts are applied in Example B of this guide.

4.1.7 Including External Files

In the process of building a workflow external files are used and it is necessary to identify them so that they are read correctly when that workflow is loaded into Inspire Designer and processed. External files include:

- Data Files
- Images
- Fonts

It is especially important to remember this behavior of workflows when moving a WFD file to a completely new directory, e.g. sending the file to somebody else.

When an external file is identified, the path to its location is detected. Therefore, if the location of the WFD file changes, the path becomes invalid. To avoid having to redefine all of the paths that have been used each time a WFD file is relocated, external files can be included to become part of the workflow. To include external files go to the menu **File | Used Files** to open the List of Used Files dialog.

The dialog lists all of the external files that are currently assigned to the workflow. The current status of the file is visible in the **State** column. The states *Exclude* and *Both* indicate that the files are not completely included in the WFD file. Further to this, from the **Load Status** column it can be determined whether the file is in the same location as it was when it was originally used in the design of the workflow. If the **Load Status** reads *OK – from HD*, then it can be included in the workflow from this dialog, but if it reads *Missing*, a new path to the file must be re-defined in the module where it has been employed.

In order to include a file to become a part of the WFD file, select it in the dialog and click on the **Include** icon. The **State** will become *Included*.

To copy the WFD file to another location and, at the same time, include all of the used files, go to the menu **File | Export with All Included**. This opens a Save As dialog allowing a copy of the file to be created in another location, but the copy will have all files included (which does not have to be the case with the original file).

4.1.8 Optimizing a Workflow

Optimization is about considering an already completed workflow and taking steps to improve it with regards to efficiency of processing. Two considerations that should be made are:

1. Deleting Unnecessary Components
2. Correcting the Processing Order

4.1.8.1 Deleting Unnecessary Components

In general, it is best to remove all unnecessary components from a workflow. This applies to unnecessary data manipulations and Layout objects that have been created but not used.

You can proceed by re-checking the modules of the workflow:

- Data Input – Are all the fields of the data files being read into the workflow actually being used in the design of the document in Layout?

  If not, then remove their corresponding variables in the workflow using a Data Transformer module.

- Data Transformers – Is every variable that is calculated or converted using a Data Transformer module actually used in the design?

  If not, redefine the variables so that they will no longer be calculated or converted.

- Layout – Is every object that you have created in Layout actually being used on the pages of the design?

  If not, in the Layout window go to menu **Edit | Find Unused Objects** and, in the **Find Unused Objects** dialog that opens, select the objects that you want to delete and click on the **Remove** button.

4.1.8.2 Correcting the Processing Order

The processing order of the pages of the design is governed by the properties of the pages in Layout. This relates to the page order and, less noticeably, the order of the objects under the node of the page on which they are located in the Layout Tree.

As a general rule, it is more efficient to have static objects closer to the page (i.e. they should be placed nearer to the node of the page under which they are grouped in the Layout Tree) and objects that contain variables further away. In practice, the order of the objects under a page node corresponds with the order in which you added objects to the page. Thus, the resulting order of objects in the Layout Tree may not be efficient for processing.

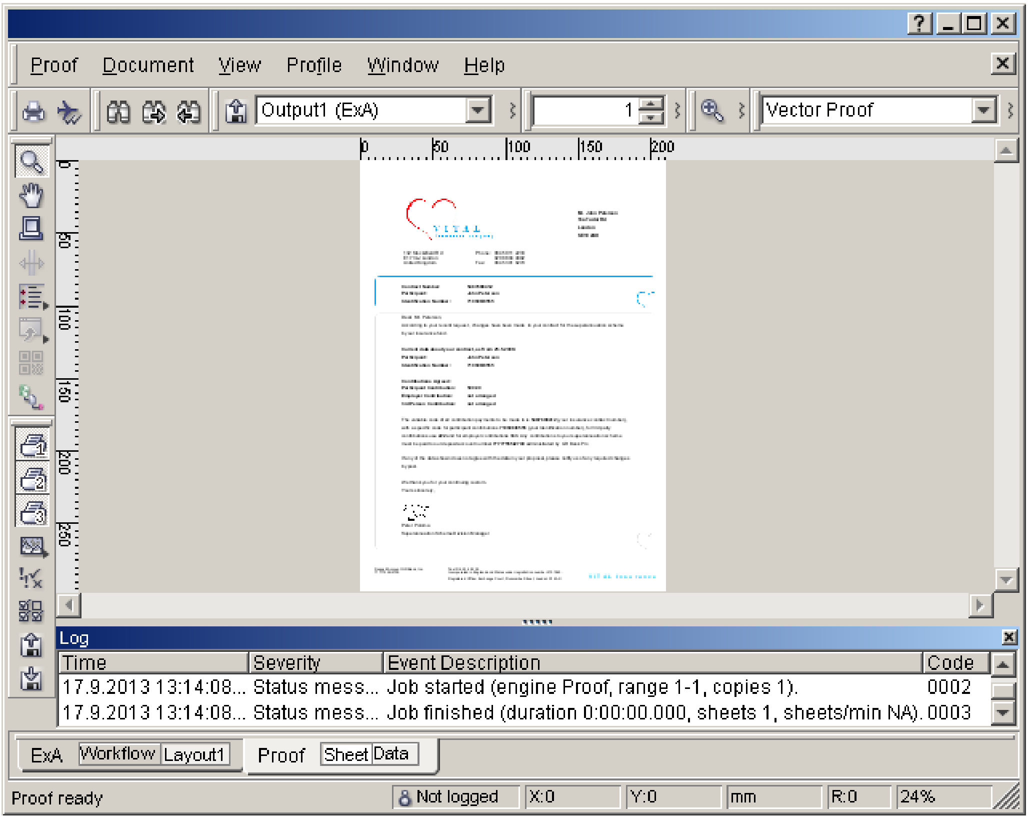
To improve the order of objects, in the Layout window, go to the menu **Edit | Optimize | Static Object Order** and all of the static objects will be moved so that they are closest to the pages.

4.2 Proofing a Workflow

The ability of Inspire Designer to process a workflow and produce a proof view provides valuable support to designing a document. It is highly recommended that you proof a workflow at regular intervals when building it, especially while designing the pages of your document in the Layout module.

To load the current workflow to proof, click on the **Proof** icon on the Workflow toolbar. Alternatively, <**F6**> is the keypad shortcut for proofing.

The Sheet Proof window opens.



Time	Severity	Event Description	Code
17.9.2013 13:14:08...	Status mess...	Job started (engine Proof, range 1-1, copies 1).	0002
17.9.2013 13:14:08...	Status mess...	Job finished (duration 0:00:00.000, sheets 1, sheets/min NA).	0003

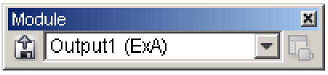
The Sheet Proof Area shows the proofed document that results from the current workflow design. It allows a review of the design with all of the required records having been processed.

As the Sheet Proof window opens, the Log window-pane can be seen to scroll as it logs events. With a simple workflow this will not take long. The details of all processing events are listed in the Log window-pane with some details about them in the **Event Description** column. If an error has occurred, it will be highlighted with a brief explanation about the problem.

If an error has occurred, return to the Workflow or Layout window to fix the problem. For scripting errors and errors in Layout, double-clicking on the error will move you to the section where the error has occurred, e.g. to a variable in Layout.

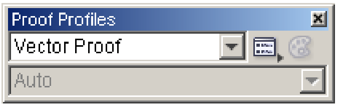
Two important features to check at this stage are:

1. The **Module** toolbar:



The combo-box on this toolbar allows the selection of the Output module via which the proof is being processed. In the case that your workflow has more than one output module, this should be checked to make sure you are proofing from the correct one. It also allows you to quickly load another proof from any other output.

2. The **Proof Profiles** toolbar:



The combo-box on this toolbar allows the selection of a proof profile. Proof profiles are pre-saved parameters for the proofing. It is possible to configure several different profiles and select between them to gain different proof results.

There are two main groups of proof profiles, vector and raster. These relate to the two different families of page description languages (vectorized and rasterized).

In addition to them, there are two more proof profiles HTML and HTML Simple, which enable you to view the content of your document as a web page or an HTML email message. In fact, what you view in Proof is an HTML document generated into the temporary folder and opened through your system web browser.

The choice of proof profile that you create and use should be according to the production engine that will be used to print the job with, i.e. vector, raster or HTML. Remember: The profile that should be used in proof can be pre-configured in the Output module.

A vector type of engine (PDF) will be employed in Example A and a raster type of engine (AFP) will be employed in Example B of this guide.

To create proof profiles, select *Create New Profile* from the combo-box to open the Create Profile dialog where you can specify the **Name** and **Type** (*Vector*, *Raster*, *HTML* or *HTML Simple*) of the new profile.

To adjust the parameters of the profile, click on the **Edit** icon to open the Edit Profile dialog for that profile type.

Once the correct profile and module are selected, you can scroll through the proofed pages to check them using the arrow icons located at the top of the window.

If there is a need to close the Proof environment, go to menu **Proof | Exit Proof**. However, the Proof environment can remain open in case it is required again.

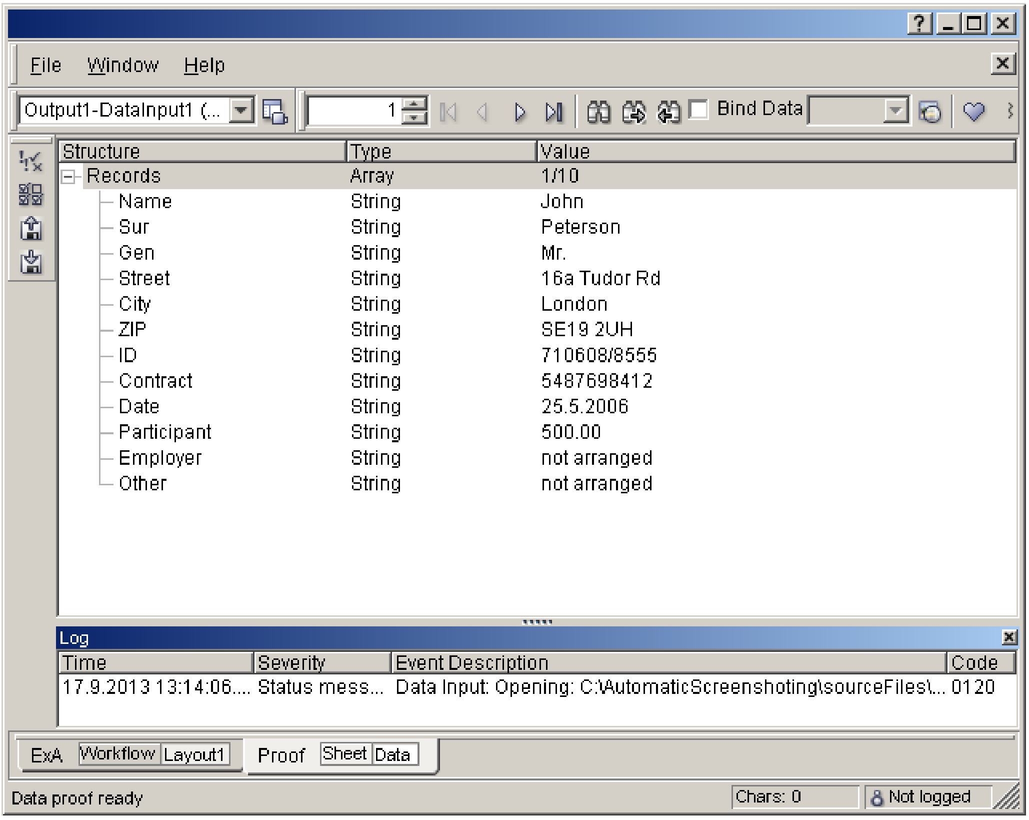
If you modify any of the settings of the workflow modules, Proof must be run again in order to obtain an updated proof of the design, as the Proof tab only shows the status of the design at the time when it was run.

Proofing will be applied in Example A and Example B of this guide.

4.2.1 Data-Proofing a Workflow

The processing of the workflow that is performed by proofing it, also provides information about how the data has been processed at different stages in the workflow. This can be checked in the Data Proof window.

Navigate to the Data Proof window using the Proof tab .

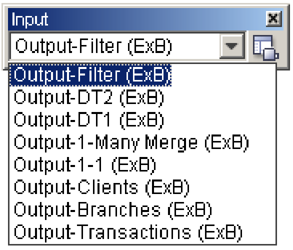


Structure	Type	Value
Records	Array	1/10
Name	String	John
Sur	String	Peterson
Gen	String	Mr.
Street	String	16a Tudor Rd
City	String	London
ZIP	String	SE19 2UH
ID	String	710608/8555
Contract	String	5487698412
Date	String	25.5.2006
Participant	String	500.00
Employer	String	not arranged
Other	String	not arranged

Time	Severity	Event Description	Code
17.9.2013 13:14:06....	Status mess...	Data Input: Opening: C:\AutomaticScreenshoting\sourceFiles\...	0120

The details of the data processing events are listed in the Log window-pane with some details about them in the **Event Description** column. If an error has occurred, it will be highlighted with a brief explanation about the problem.

An important feature to check at this stage is the **Input** toolbar in the top left corner:



The combo-box on this toolbar allows the selection of the Output module via which the proof is being processed and the point in the workflow that the data is relevant to, i.e. different stages of data processing. This is particularly helpful if you have a workflow with many data manipulations and you want to check how the data has been processed in each module of the workflow.

Once the correct module is selected, you can scroll through the proofed records to check them using arrow icons located at the top of the window.

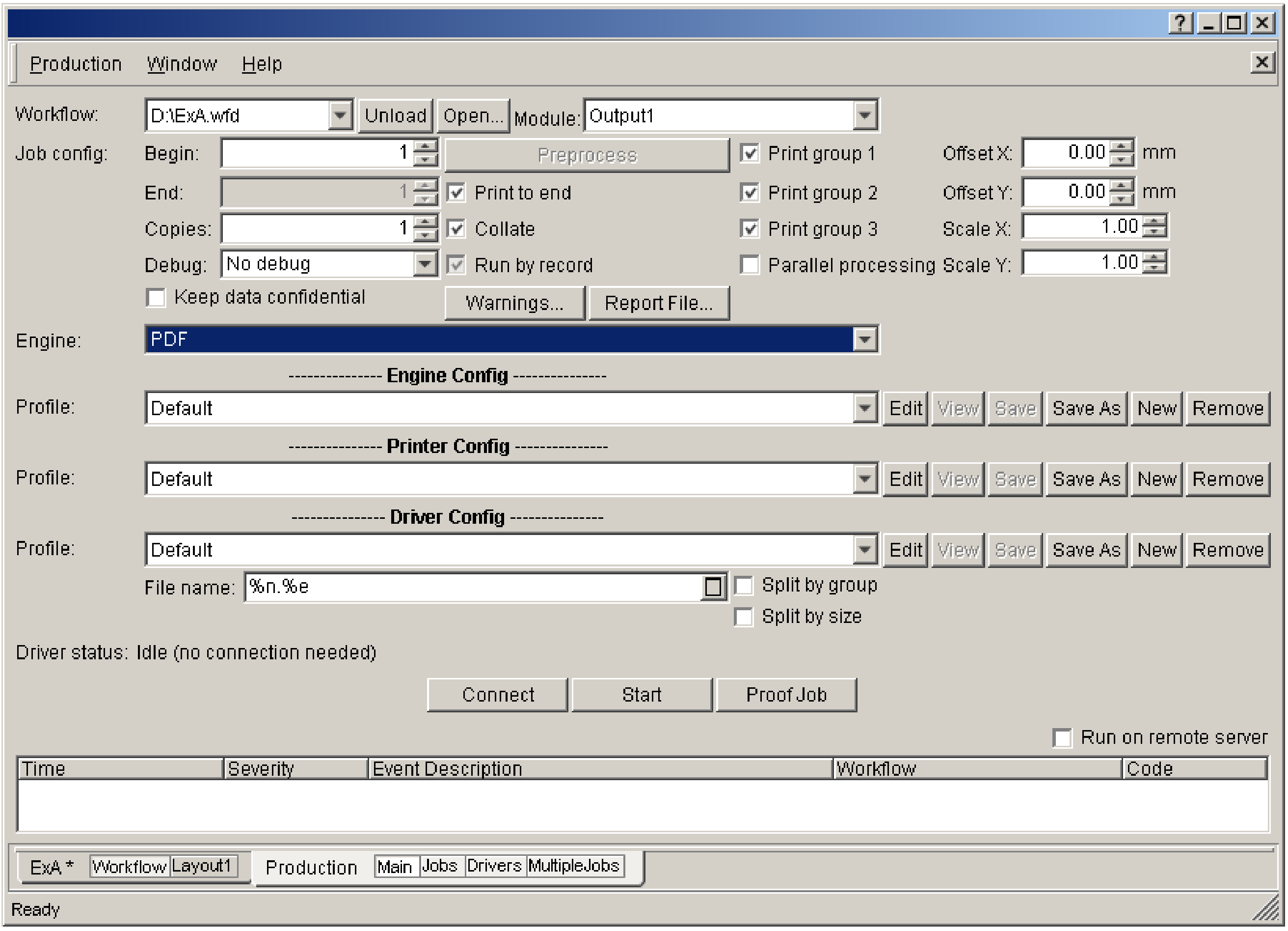
Data Proofing is applied in Example B of this guide.

4.3 Producing a Document

The Production environment of Inspire Designer acts a separate application to which different workflow files (WFD) can be loaded and, with some additional parameters being set, processed to a printing device for printing.

In order to produce the workflow that is currently open and selected in Inspire Designer, go to the Workflow window and click on the **Production** icon on the Workflow toolbar. Alternatively, <**F5**> is the keypad shortcut for loading a workflow to production.

The Production environment opens with the Main Production window.



Workflow: D:\ExA.wfd

Module: Output1

Job config:

Begin: 1

End: 1

Copies: 1

Debug: No debug

☑ Print to end

☑ Collate

☑ Run by record

☑ Print group 1

☑ Print group 2

☑ Print group 3

□ Parallel processing

Offset X: 0.00 mm

Offset Y: 0.00 mm

Scale X: 1.00

Scale Y: 1.00

□ Keep data confidential

Engine: PDF

Profile: Default

Profile: Default

Profile: Default

File name: %n.%e

□ Split by group

□ Split by size

Driver status: Idle (no connection needed)

□ Run on remote server

Time	Severity	Event Description	Workflow	Code

Ready

The **Workflow** combo-box at the top of the window shows which workflow is loaded and can be used to select between other open workflows.

Other workflow settings are:

- **Module** – A combo-box that allows the selection of the Output module via which the production should be processed. In the case that your workflow has more than one module, this should be checked to make sure you are printing from the correct one. It also allows you to quickly load any other output.

- **Job Config** – Edit-boxes and check-boxes (**Begin**, **End**, **Print to End**, **Scale** etc.) that control the selection of which pages of the design will be sent to the printer and allow the adjustment of basic presentation characteristics.

Once it is ensured that the correct workflow and module are loaded (and that the job configuration is as required) the steps for production are:

1. Select a page definition language from the **Engine** combo-box. It should correspond with the printer you will send the print job to.

   Remember: The engine that should be used in production can be pre-configured in the Output module.

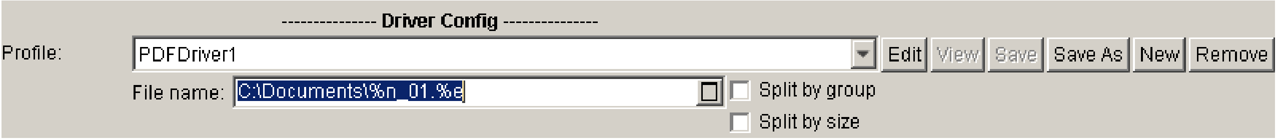
2. Create an engine config profile by clicking on the **New** button in the **Engine Config** options and make sure it is selected in the **Profile** combo-box.

   Edit the engine profile to suit the needs of your job by clicking on the **Edit** button to open the configuration dialog.

3. Also, you can create a driver config profile by clicking on the **New** button in the **Driver Config** options. There are many possible types and the selection available depends on the selected engine type. In many cases *Spool file* is used. This produces a file that can be sent to the printer later.

   After the driver profile is created make sure it is selected in the **Profile** combo-box.

4. Give the file to be printed a name and enter a path to the location it should be written to:



-------------- Driver Config --------------

Profile: PDFDriver1

File name: C:\Documents\%n_01.%e

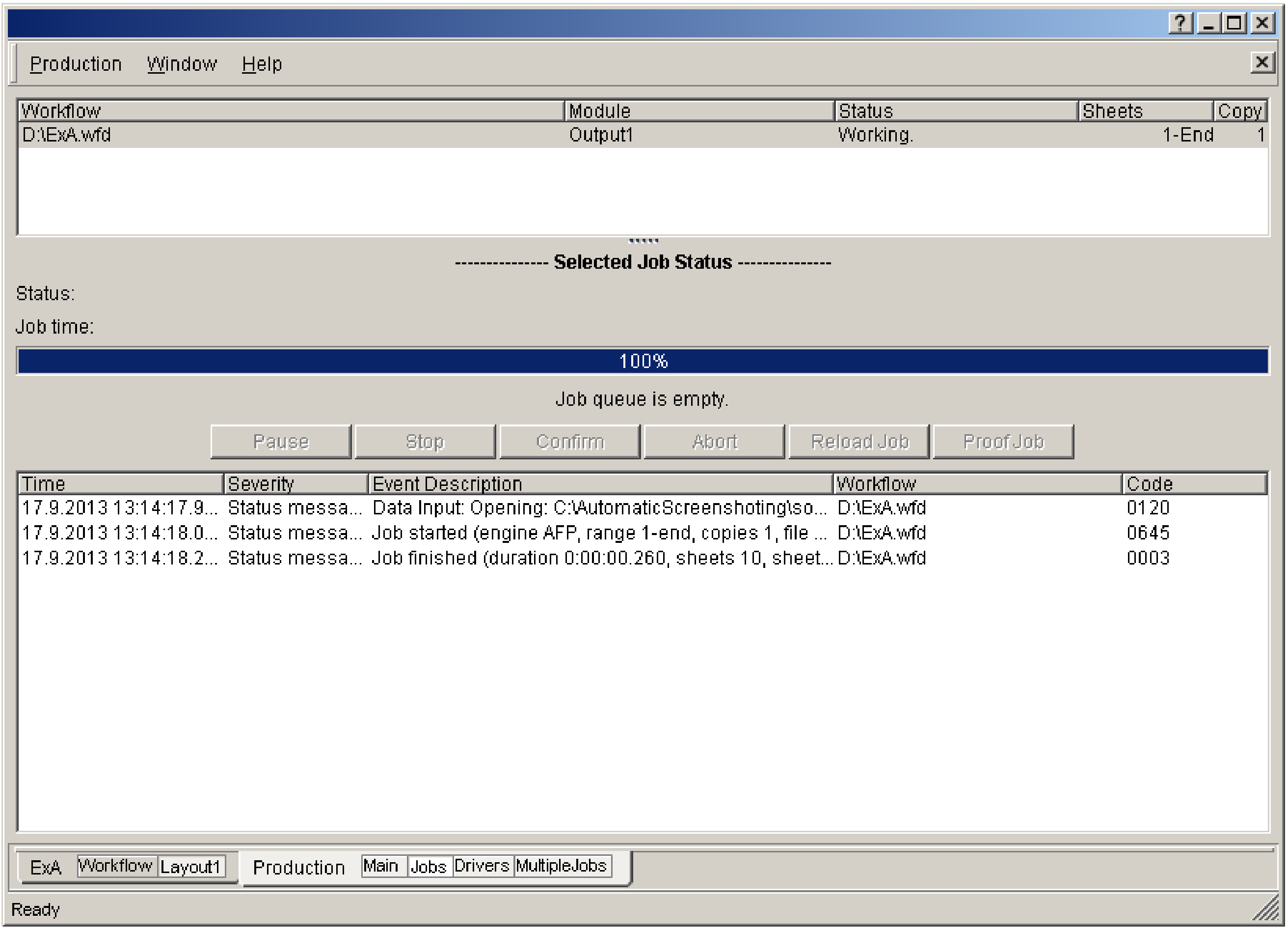
□ Split by group

□ Split by size

   There are some auto-naming formulae available that create a file name based on workflow parameters (e.g. *%n* represents the name of the workflow file; *%e* represents the default extension of the engine type selected).

5. Click on **Start** to print the workflow, i.e. in many cases to create a spool file to the chosen location.

The Jobs window then opens:



Workflow	Module	Status	Sheets	Copy
D:\ExA.wfd	Output1	Working.	1-End	1

-------------- Selected Job Status --------------

Status:

Job time:

100%

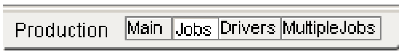
Job queue is empty.

Time	Severity	Event Description	Workflow	Code
17.9.2013 13:14:17.9...	Status messa...	Data Input: Opening: C:\AutomaticScreenshoting\so...	D:\ExA.wfd	0120
17.9.2013 13:14:18.0...	Status messa...	Job started (engine AFP, range 1-end, copies 1, file ...	D:\ExA.wfd	0645
17.9.2013 13:14:18.2...	Status messa...	Job finished (duration 0:00:00.260, sheets 10, sheet...	D:\ExA.wfd	0003

As the Jobs window opens, the Log area at the bottom reports the events of production (spooling). The details of all processing events are listed in the Log area with some details about them in the **Event Description** column. If an error has occurred it will be highlighted with a brief explanation about the problem.

The Workflow area at the top of the window lists all jobs that have been run during this production session until they have been dismissed from the history.

To return to the Main Production window use the production tab:



> **NOTE** For some engine types it is also possible to create a Printer Config profile. If so, the extra **Profile** combo-box will be shown in the Production window.

Production and engines will be used in Example A and Example B of this guide.

5 Examples

The term 'variable-data printing' covers the broad-base of clients that Inspire Designer is geared towards. This can mean printing mail to customers/clients which is at least partially dependent on personalized information which is stored in databases. Other typical cases can be transactional jobs such as bank account statements; again with variable data, which is different for each of the clients.

In this chapter we will create two example workflow files, proofing and printing them as we go. Each of the examples is taken from a slightly different market where Inspire Designer is being used and shows a different level of complexity in the workflow created. By working through the examples you will be progressively introduced to increasingly advanced features of Inspire Designer while being informed about its practical application.

The examples use various input types in differing designs and produce different output types. In this way it is hoped that the examples are more universal, if one example does not suit an input, design or output format that you recognize, then look in the other example for a familiar format and substitute the particular set of instructions that apply.

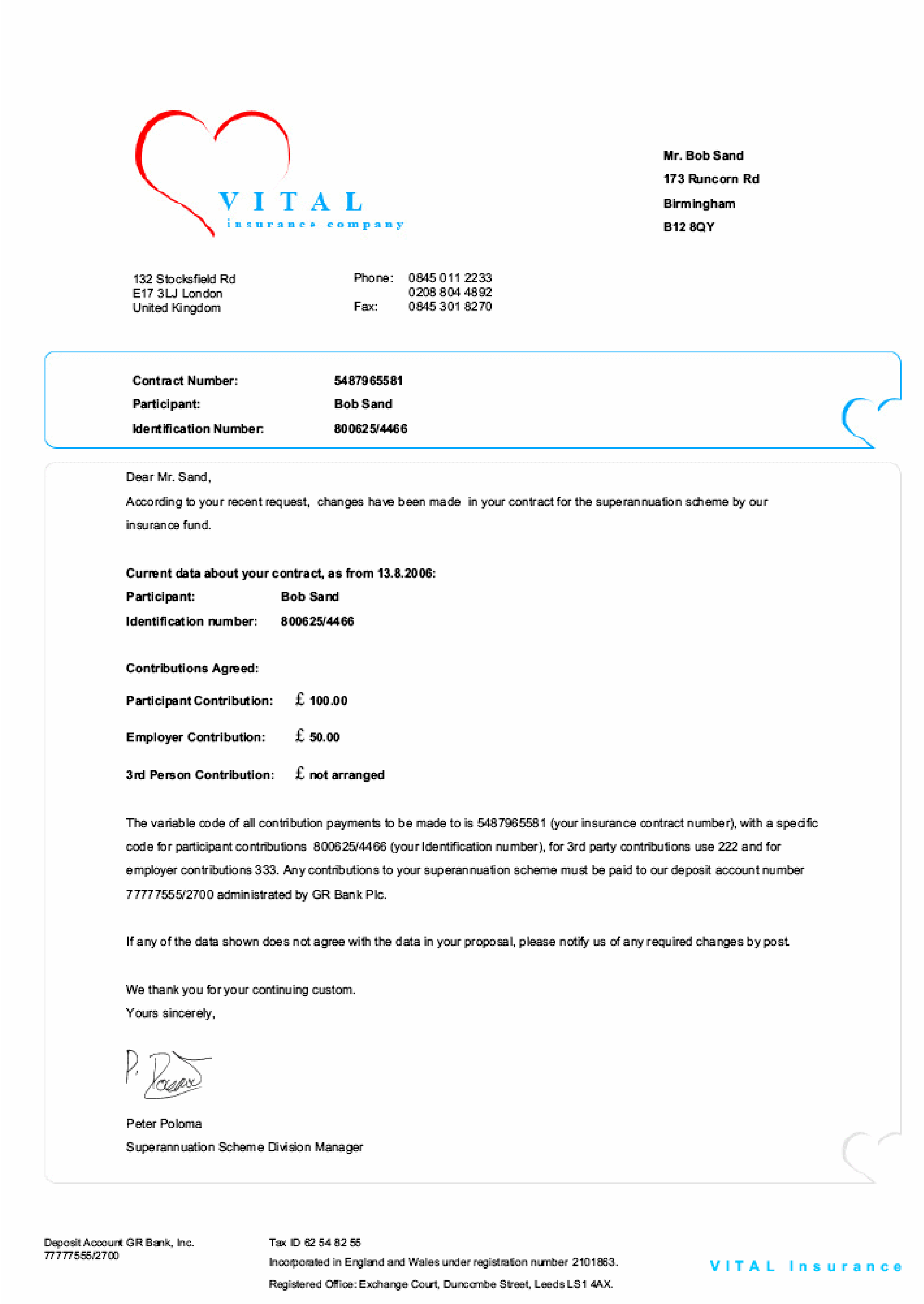
Specific input/output formats described in the two examples are:

- Example A – Personalized Letter
    - Input – Comma Separated Value (CSV) data file.
    - Output – Portable Document Format (PDF) file.
- Example B – Account Statement
    - Input – Comma Separated Value (CSV), Fixed Length Fields (TXT) and XML formatted data files.
    - Output – Advanced Function Presentation (AFP) spool file.

5.1 Example A – Personalized Letter

One common use for Inspire Designer is in the insurance industry. Such companies have many clients, each of whom have taken out different insurance policies.

As a scenario for this example, all of the clients' information is stored in a database and when changes in the database are made it is required to inform those clients affected about them. As a mail to convey the changes, a standard letter is designed that covers the issue and, because each person has different data in their records, personal details need to be written directly from the database to specified positions on the letter, e.g. a person's name needs to be written in the address area.



VITAL
insurance company

Mr. Bob Sand
173 Runcorn Rd
Birmingham
B12 8QY

132 Stocksfield Rd
E17 3LJ London
United Kingdom

Phone:	0845 011 2233 0208 804 4892
Fax:	0845 301 8270

Contract Number:	**5487965581**
Participant:	**Bob Sand**
Identification Number:	**800625/4466**

Dear Mr. Sand,

According to your recent request, changes have been made in your contract for the superannuation scheme by our insurance fund.

Current data about your contract, as from 13.8.2006:

Participant:	**Bob Sand**
Identification number:	**800625/4466**

Contributions Agreed:

Participant Contribution:	£ **100.00**
Employer Contribution:	£ **50.00**
3rd Person Contribution:	£ **not arranged**

The variable code of all contribution payments to be made to is 5487965581 (your insurance contract number), with a specific code for participant contributions 800625/4466 (your Identification number), for 3rd party contributions use 222 and for employer contributions 333. Any contributions to your superannuation scheme must be paid to our deposit account number 77777555/2700 administrated by GR Bank Plc.

If any of the data shown does not agree with the data in your proposal, please notify us of any required changes by post.

We thank you for your continuing custom.

Yours sincerely,

P.

Peter Poloma
Superannuation Scheme Division Manager

Deposit Account GR Bank, Inc.
77777555/2700

Tax ID 62 54 82 55
Incorporated in England and Wales under registration number 2101863.
Registered Office: Exchange Court, Duncombe Street, Leeds LS1 4AX.

VITAL Insurance

Figure 5.1 An Example Design for the Personalized Letter

This personalized letter consists of the following elements:

1. One Page (design for each record)

2. Logo and Background Images (consists of static data only)

3. Company Details Text (consists of static data only)

4. Client Address Area (consists of variable data and static text)
5. Client Details Area (consists of variable data and static text)
6. Personalized Letter Body (consists of variable data and static text)

Normal practice would be to have template details (2 and 3) pre-printed onto the media and the personalized details (4, 5 and 6) only to be added by the workflow, but for the purpose of this exercise it is instructive to add all of the features as if to print on a blank media.

The function of the workflow we create will be to:

- Read from a database some records of clients that do not require any conversion.
- Arrange different fields from the records to a simple, one-page-letter design.
- Print each letter on a single page of a PDF document that has bookmarks to identify each client's letter.

In this example, the workflow to perform these functions will be composed of three modules:

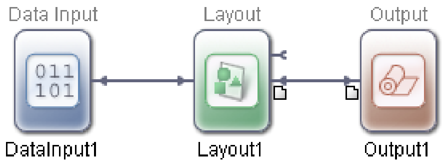


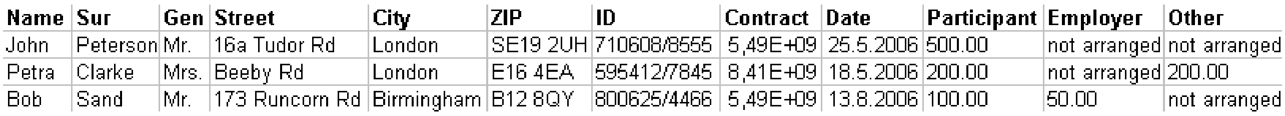
Figure 5.2 The Modules of Example A

5.1.1 Requirements

To complete Example A using Inspire Designer, some external files are required so that they can be read or inserted into the workflow to become objects that appear in the letter:

- A data file that contains client details that should be three or more records long. All fields of this data file should be entered in string data-type, i.e. text that will be comprehensible in its present format when printed to the letter.

  To compile the example, we have used a Comma Separated Value (CSV) file with the following fields: Name, Sur (the client's surname), Gen (the gender-dependent salutation for the client), Street, City, ZIP (the postal code), ID (an identification number), ContractNr (an insurance number), Date (the date on which the changes were made), Participant (represents a cash amount paid by the client), Employer (represents a cash amount paid by the client's employer) and Other (represents a cash amount paid by a third party):



Name	Sur	Gen	Street	City	ZIP	ID	Contract	Date	Participant	Employer	Other
John	Peterson	Mr.	16a Tudor Rd	London	SE19 2UH	710608/8555	5,49E+09	25.5.2006	500.00	not arranged	not arranged
Petra	Clarke	Mrs.	Beeby Rd	London	E16 4EA	595412/7845	8,41E+09	18.5.2006	200.00	not arranged	200.00
Bob	Sand	Mr.	173 Runcorn Rd	Birmingham	B12 8QY	800625/4466	5,49E+09	13.8.2006	100.00	50.00	not arranged

The first three records of the CSV file used to compile Example A

- Some image files (e.g. JPG and PCX files) to be used for the look of the letter: One logo and two background images (that can be written over without obscuring the text) and an image of a scanned signature.

5.1.2 Workflow

⊕ Open the Workflow window where a simple workflow can be designed to create a personalized letter.

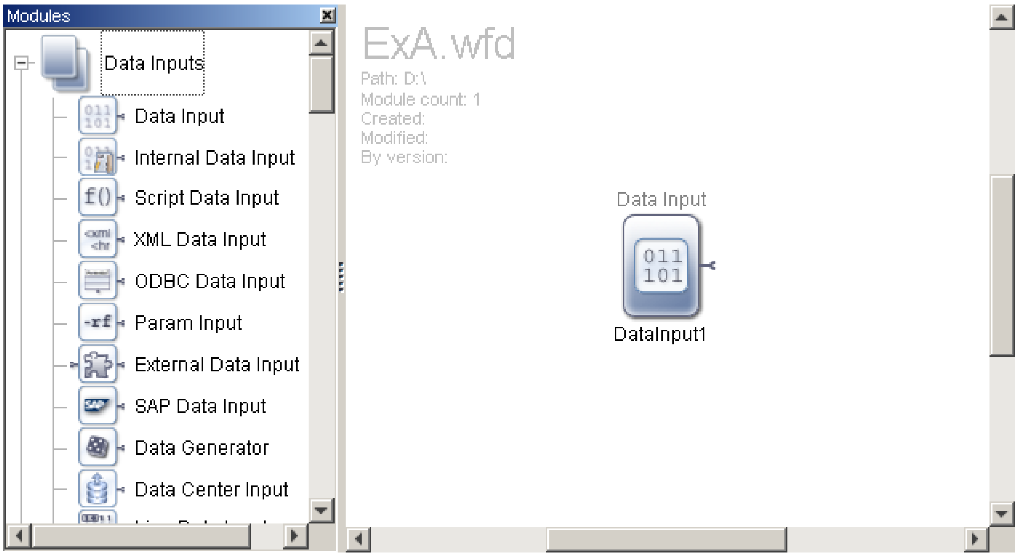
Start Inspire Designer and then select **New Workflow** icon from the Startup dialog to open the Workflow window.

Remember [16]: Save the workflow with a new name in an easy-to-find location (e.g. ExA.wfd in the same directory as the images and data file that you will use).

5.1.2.1 Data Input

⊕ Add a Data Input module that can read the format of the data in the CSV file.

Select Data Input from the Module Tree, drag it to the Workflow Area and drop it.



Path: D:\
Module count: 1
Created:
Modified:
By version:

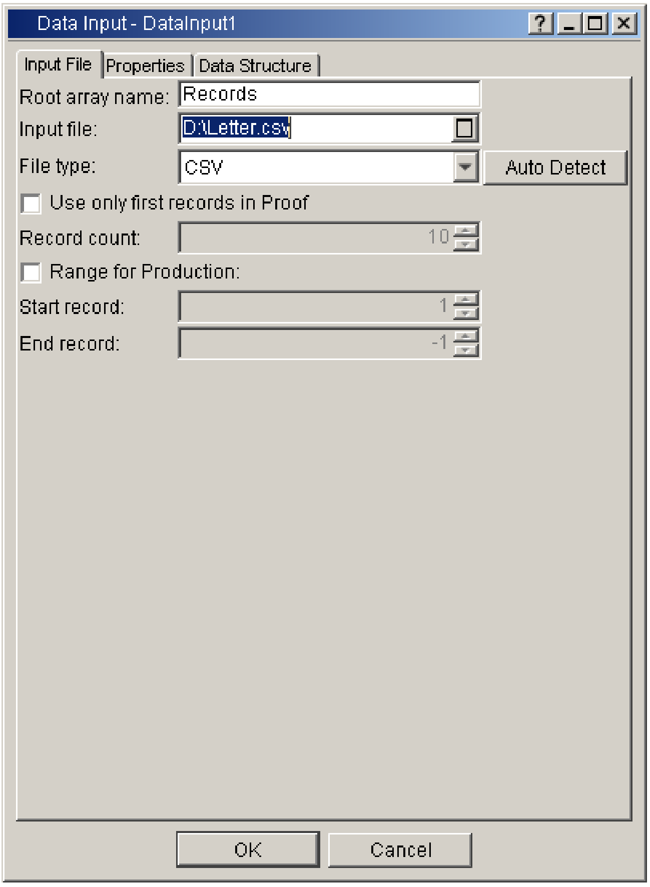
⊕ Configure the data module to be able to read the data file.

Double-click on the Data module to open it.

⊕ Enter the path to the file.

In the **Input file** edit-box, browse □ for the CSV file (e.g. Letter.csv) and click on **Open**.

The full path to the file is displayed.



Root array name: Records

Input file: D:\Letter.csv

File type: CSV

□ Use only first records in Proof

Record count: 10

□ Range for Production:

Start record: 1

End record: -1

⊕ Set other attributes to read the data file correctly.

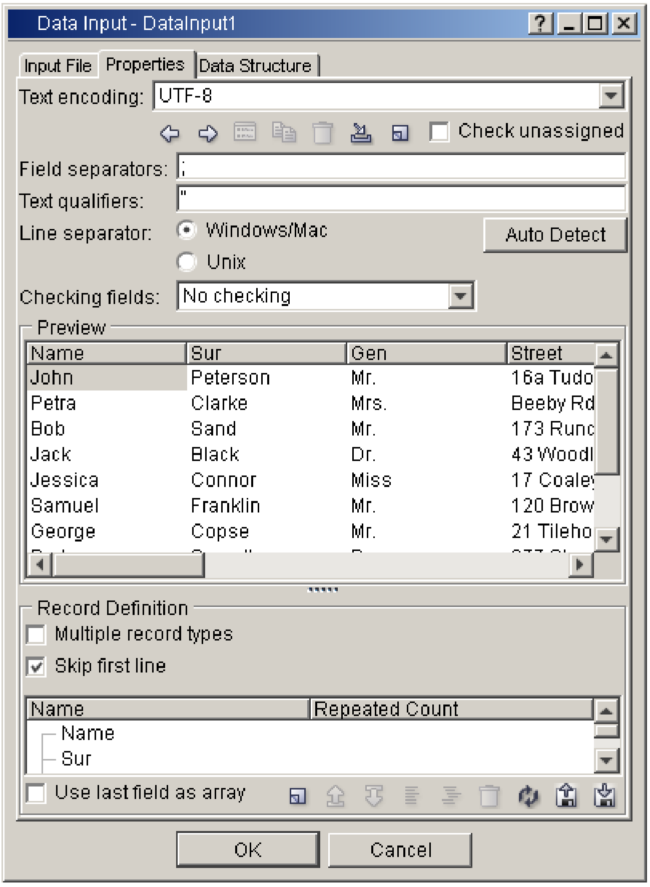
Check that the attribute **File type** matches the file used (*CSV* is the default setting).

All attributes on the Input File tab are set.

⊕ View the file-type specific parameters.

Go to the Properties tab.

The attributes of the CSV file are displayed:



Data Input - DataInput1

Input File | Properties | Data Structure

Text encoding: UTF-8

□ Check unassigned

Field separators: ;

Text qualifiers: "

Line separator: ● Windows/Mac ○ Unix

Auto Detect

Checking fields: No checking

Preview

Name	Sur	Gen	Street
John	Peterson	Mr.	16a Tudo
Petra	Clarke	Mrs.	Beeby Rd
Bob	Sand	Mr.	173 Runc
Jack	Black	Dr.	43 Woodl
Jessica	Connor	Miss	17 Coaley
Samuel	Franklin	Mr.	120 Brow
George	Copse	Mr.	21 Tileho

Record Definition

□ Multiple record types

☑ Skip first line

Name	Repeated Count
Name	
Sur	

□ Use last field as array

OK | Cancel

The **Text encoding** combo-box selects the codec to be used for the text. Be careful that the encoding that you select recognizes all of the characters from your data file. The attributes **Field separator**, **Text qualifiers** and **Line separator** are set with default values. These values must match the actual format of the data file that is used. If your data file does not have the same parameters the records will not show correctly in the **Preview** area.

⊕ Check the attributes to see that they match the format of the data file.

Click on the **AutoDetect** button in order to correctly fill these attributes.

All attributes on the Properties tab are set.

⊕ Confirm the configuration and return to the Workflow window.

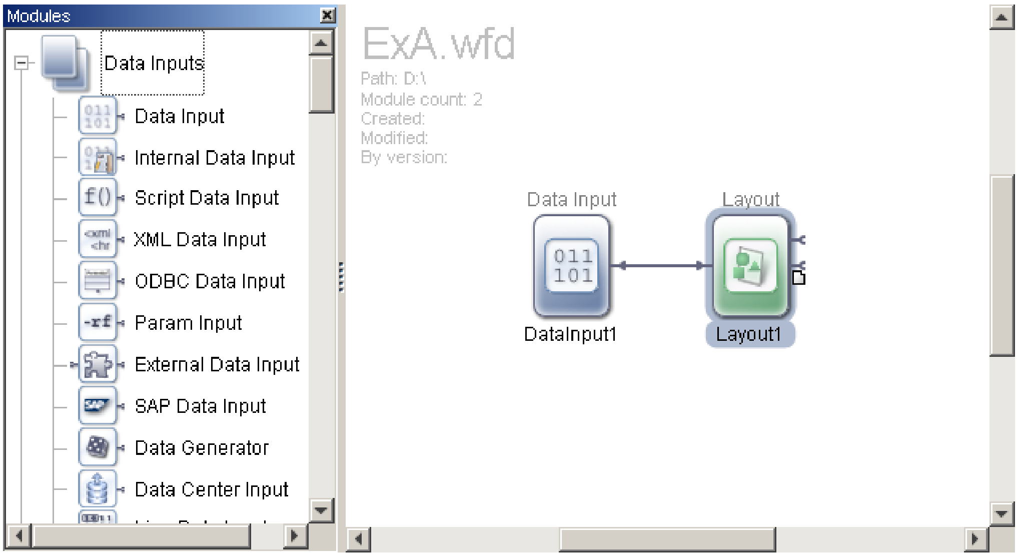
Click on **OK**.

The Data Input module closes.

5.1.2.2 Layout

⊕ Add a Layout module in which to design the page of the letter.

Select Layout from the Module Tree and drag it to the Workflow Area. Then drop it and connect it to the Data Input module.



ExA.wfd
Path: D:\
Module count: 2
Created:
Modified:
By version:

⊕ Start the design of the letter that will be personalized and sent to each of the clients who's details make up the records of the CSV file.

Double-click on the Layout module to open it.

Remember [46]: The letter being created will consist of the following elements:

1. One Page
2. Logo and Background Images
3. Company Details Text
4. Client Address Area
5. Client Details Area
6. Personalized Letter Body

One Page

When the Layout module opens, a new page is displayed in the Layout Area. This page (by default: Page1) can be seen in the Layout Tree when you expand the Pages family.

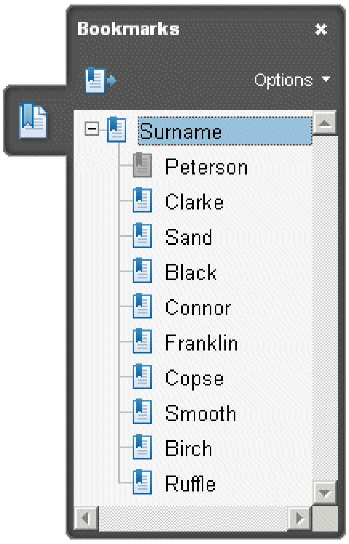
This is the only page within the layout and will require only one type of media to print on. Therefore, there is not really any need to set the sheet name 1 that could be used to identify the page later in the workflow. However, in order to get familiar with the concept of sheet names, we will assign a sheet name to this page.

⊕ Set up an identifier for the page.

Select the page (e.g. Page1) in the Layout Tree and go to its properties panel. Select the Sheet Names tab and enter a name (e.g. *Letter*) for sheet name 1 into the first row to the **Constant Value** field.

Remember: Further to this sheet name 1, other sheet names can be assigned. In our example, it would be helpful to have a sheet name that is related to the records that are read from the data input, so that each record that is read from the data file returns a new value for the sheet name. Then, our 1 page design could print 1 sheet for each of 10 clients producing a total of 10 sheets. Because we used this record related sheet name, we can differentiate between each of the 10 sheets. Therefore we will assign the surname of the client that the sheet was printed for (i.e. the Sur variable) to sheet name 2.

Later in this example, we will use this sheet name returning the surnames to create bookmarks for the PDF document that we will print.



Surname
Peterson
Clarke
Sand
Black
Connor
Franklin
Copse
Smooth
Birch
Ruffle

Figure 5.3 An Example of Bookmarks Created using Sheet Names (Viewed in Adobe Reader)

⊕ Set up an index for the sheets to be printed, so that it can be used to create a bookmark for each client when a PDF file is produced.

In the page's properties panel go to the Sheet Names tab and select the client surname variable (e.g. *Sur*) from the **Variable** combo-box of the second row.

This variable returns the client's surname for each letter that is printed.

Remember: Rename the page (e.g. to "Letter") for a better overview.

Logo and Background Images (2)

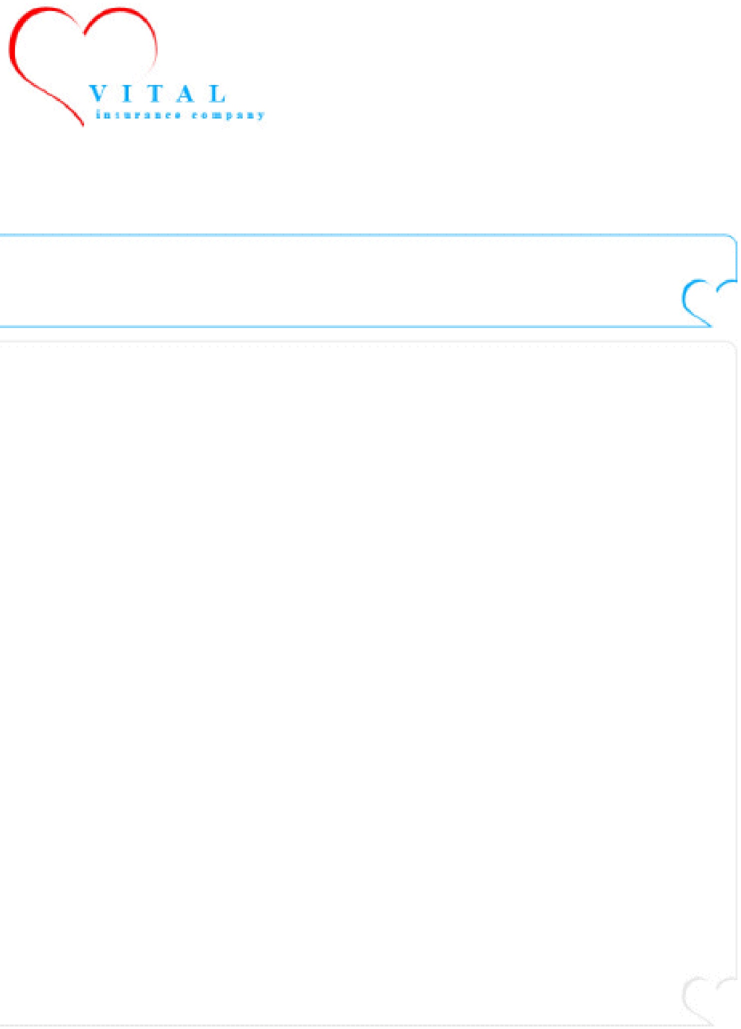


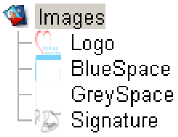
Figure 5.4 An Example of Logo and Background Images

Most of the objects that are needed to design this example can be created using the features and tools of the Layout environment. However, images are external files that need to be located so that they can be used.

⊕ Link to the images that will be used for the letter's design.

Go to the menu **Insert | Insert Image**. In the standard Open dialog that opens, browse for the images, select all of them and click on **Open**.

The images appear as objects in the Layout Tree (under the Images family).



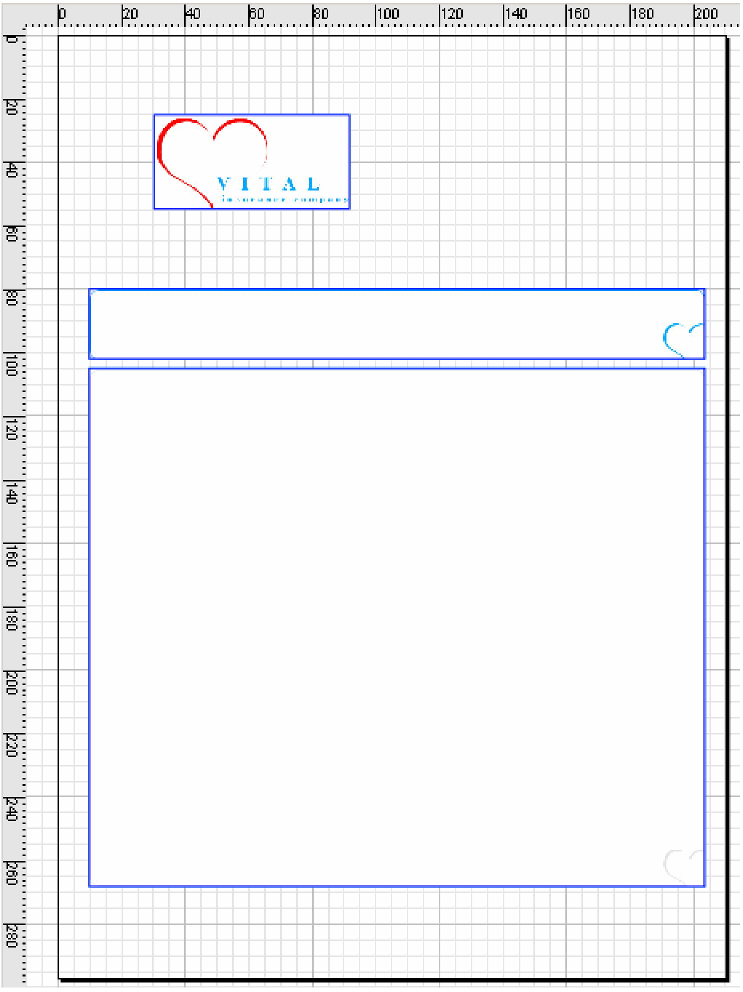
Images
- Logo
- BlueSpace
- GreySpace
- Signature

The images are named, by default, using the names of the files that were inserted. We will use some of these images as a background for the letter.

⊕ Arrange the background images on the page.

Select the relevant images for the background (e.g. Logo, BlueSpace and GraySpace) from the Layout Tree and drag them into position, using the following screenshot for guidance.

The background images are placed on the page using image areas to define their position:



Remember: Rename the image areas (which are grouped under the node of the page onto which they have been placed).

Remember: The dependency of objects can be visualized in the Layout Tree. In our example, we can see that the Pages family contains our page (e.g. Letter) which contains our image areas each of which, in turn, contains our images (e.g. Logo, BlueSpace and GraySpace).

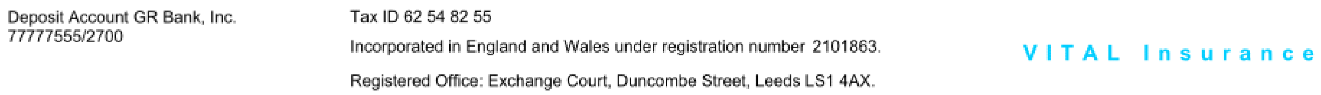
Remember: You can proof your work at any time to check that it would print how you expect it to and then, if you are happy with the result, save the workflow.

Company Details Areas (3)

132 Stocksfield Rd
E17 3LJ London
United Kingdom

Phone: 0845 011 2233
0208 804 4892
Fax: 0845 301 8270

Figure 5.5 An Example of Company Details Areas – At the Top of the Page



Deposit Account GR Bank, Inc.
77777555/2700

Tax ID 62 54 82 55
Incorporated in England and Wales under registration number 2101863.
Registered Office: Exchange Court, Duncombe Street, Leeds LS1 4AX.

VITAL Insurance

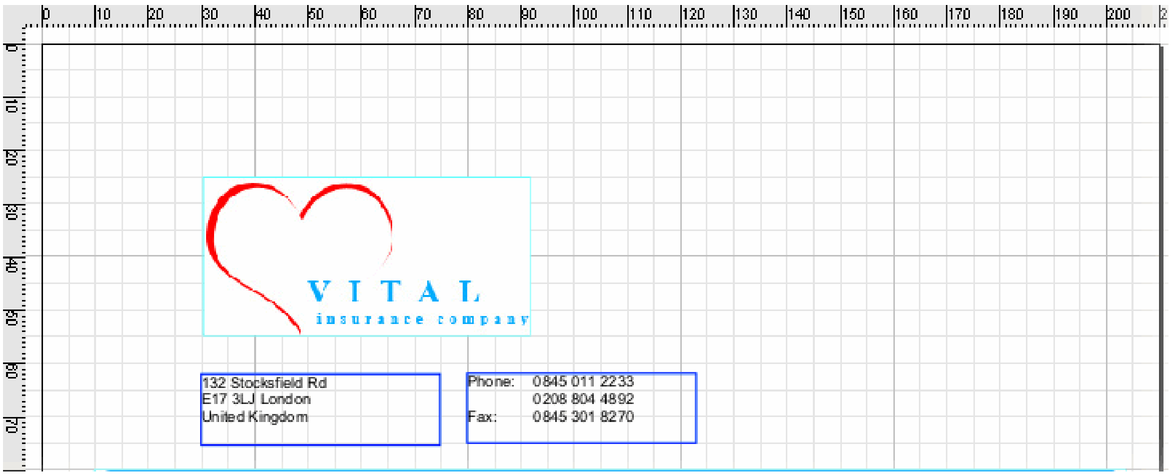
Figure 5.6 An Example of Company Details Areas – At the Bottom of the Page

The company details areas are five flow areas with simple 'static' text inside. In our example, two of the flow areas will be placed at the top of the page and three flow areas will be at the bottom. The text can also be formatted using styles and fonts via the Text Format toolbar at the top of the Layout window.

⊕ Add static text relating to company address and details to the page to complete the template.

Using the **Flow Area Tool**, draw flow areas on the page in the Layout Area. Then, edit the flows that are within them so that they contain 'company detail' text. Use the following screenshot for guidance.

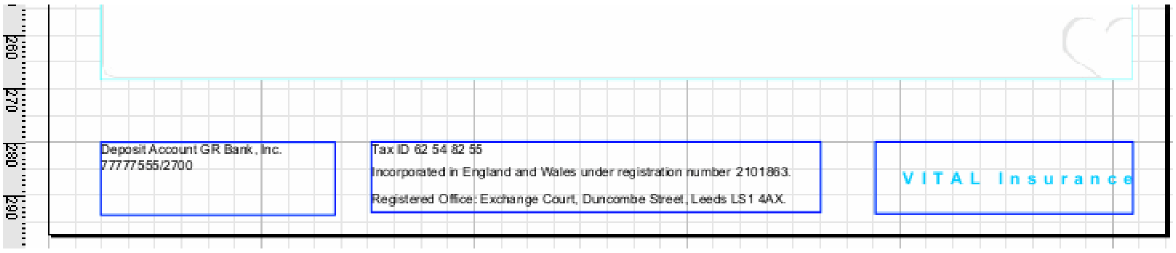
Flow areas containing static text are placed as part of the letter:



132 Stocksfield Rd
E17 3LJ London
United Kingdom

Phone: 0845 011 2233
0208 804 4892
Fax: 0845 301 8270

Top of the Page



Deposit Account GR Bank, Inc.
77777555/2700

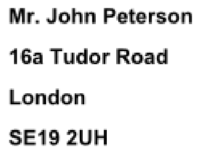
Tax ID 62 54 82 55
Incorporated in England and Wales under registration number 2101863.
Registered Office: Exchange Court, Duncombe Street, Leeds LS1 4AX.

Bottom of the Page

Remember: Rename the company details flow areas and the flows within them.

Remember: You can proof your work at any time to check that it would print how you expect it to and then, if you are happy with the result save the workflow.

Client Address Area (4)



Mr. John Peterson
16a Tudor Road
London
SE19 2UH

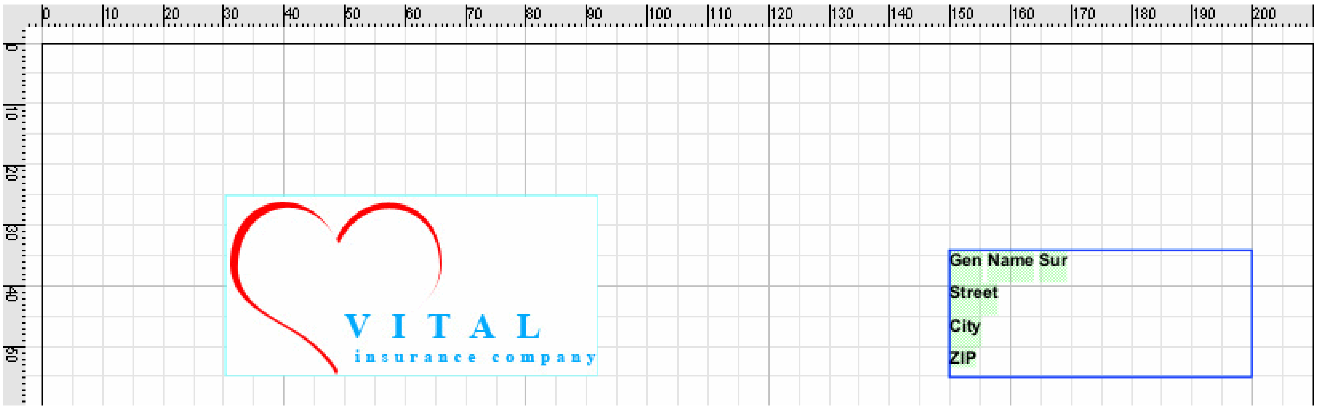
Figure 5.7 An Example of a Client Address Area

The Client Address Area is a flow area that contains only data that is read from the records of the data file specified in the data input module, i.e. variables. This text can also be formatted using styles and fonts using the Text Format toolbar at the top of the Layout window.

⊕ Add variables to the page so that an individual client's address will be printed on the letter.

Draw another flow area and, with the **Flow Area Tool** ▤ still selected, drag and drop address related variables from the Layout Tree (Path: Data/Records/Value) using the following screenshot for guidance.

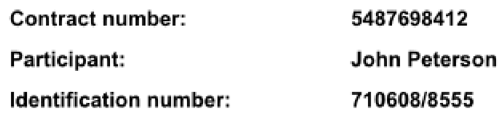
A flow area that will contain the addresses of the clients (as read from the variables) is created:



Remember: Rename the client address flow area and the flow within it.

Remember: You can proof your work at any time to check that it would print how you expect it to and then, if you are happy with the result save the workflow.

Client Details Area (5)



Contract number:	**5487698412**
Participant:	**John Peterson**
Identification number:	**710608/8555**

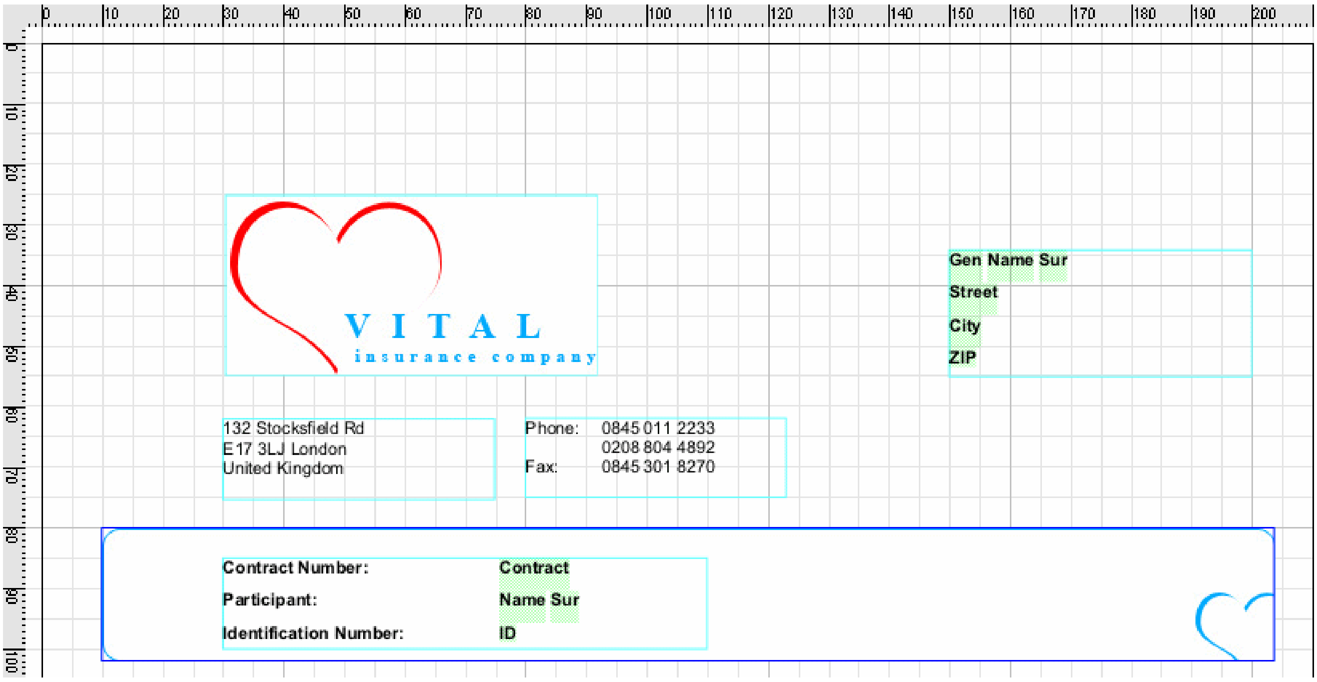
Figure 5.8 An Example of a Client Details Area

The Client Details Area is a flow area with both static text (that will appear on every letter) and variables that read the corresponding client's details (that will appear on individual letters). The text can also be formatted using styles and fonts by employing the Text Format toolbar at the top of the Layout window.

⊕ Add static text (the same for every client) next to variables (individual to each client) to the page to display some more client details inside one of the background images.

Draw a flow area over the image area, with the **Flow Area Tool** selected, type text and drag and drop client detail related variables from the Layout Tree. Use the following screenshot for guidance.

A flow area containing variable customer details is displayed within the borders of the image:



Gen Name Sur
Street
City
ZIP

132 Stocksfield Rd
E17 3LJ London
United Kingdom

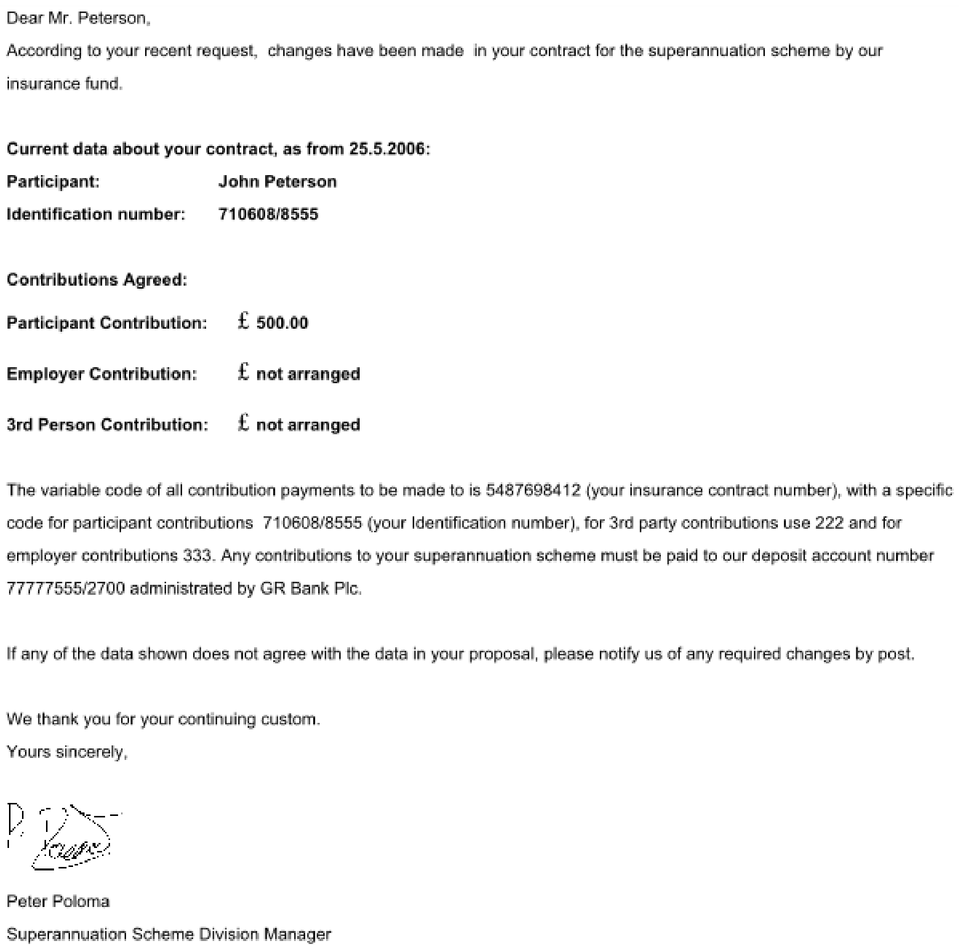
Phone: 0845 011 2233
0208 804 4892
Fax: 0845 301 8270

Contract Number: Contract
Participant: Name Sur
Identification Number: ID

Remember: Rename the client details flow area and the flow within it.

Remember: You can proof your work at any time to check that it would print how you expect it to and then, if you are happy with the result save the workflow.

Personalized Letter Body (6)



Dear Mr. Peterson,

According to your recent request, changes have been made in your contract for the superannuation scheme by our insurance fund.

Current data about your contract, as from 25.5.2006:

Participant: **John Peterson**

Identification number: **710608/8555**

Contributions Agreed:

Participant Contribution: £ **500.00**

Employer Contribution: £ **not arranged**

3rd Person Contribution: £ **not arranged**

The variable code of all contribution payments to be made to is 5487698412 (your insurance contract number), with a specific code for participant contributions 710608/8555 (your Identification number), for 3rd party contributions use 222 and for employer contributions 333. Any contributions to your superannuation scheme must be paid to our deposit account number 77777555/2700 administrated by GR Bank Plc.

If any of the data shown does not agree with the data in your proposal, please notify us of any required changes by post.

We thank you for your continuing custom.

Yours sincerely,

Peter Poloma

Superannuation Scheme Division Manager

Figure 5.9 An Example of a Personalized Letter Body

The personalized letter body contains both static text and variables. The static text is general and can be seen on all letters. Variables are placed within the static text so that each letter addresses a different client.

The text can also be formatted using styles and fonts using the Text Format toolbar at the top of the Layout window.

⊕ Add variables into static text sentences that will appear inside another of the background images, to make up the main body of the letter.

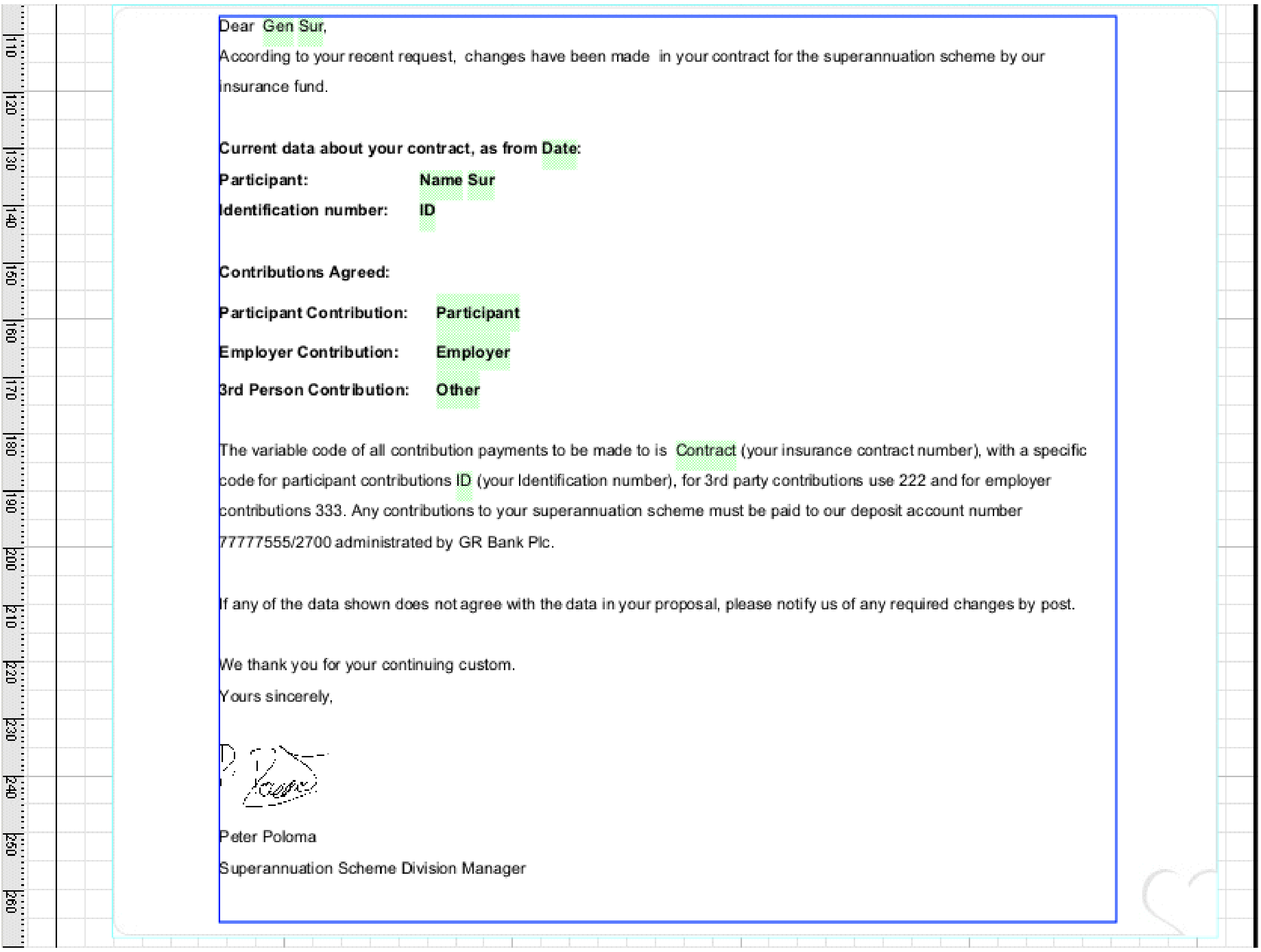
Draw a flow area over the whole bottom image area and type the text of the letter, leaving spaces where variables are wanted. Then drag and drop suitable variables located under the Data node in the Layout Tree into those spaces.

The text and variables are placed within the flow area that will write over the image.

⊕ Include the image of the signature within the text of the closing sentences of the letter.

With the **Flow Area Tool** ▤ still selected, drag the signature image (e.g. Signature) from the Layout Tree and drop it to a place in the flow. Use the following screenshot for guidance.

A flow area containing the body of the personalized letter is displayed within the borders of the other image:



Dear Gen Sur,

According to your recent request, changes have been made in your contract for the superannuation scheme by our insurance fund.

Current data about your contract, as from Date:

Participant:	**Name Sur**
Identification number:	**ID**

Contributions Agreed:

Participant Contribution:	**Participant**
Employer Contribution:	**Employer**
3rd Person Contribution:	**Other**

The variable code of all contribution payments to be made to is Contract (your insurance contract number), with a specific code for participant contributions ID (your Identification number), for 3rd party contributions use 222 and for employer contributions 333. Any contributions to your superannuation scheme must be paid to our deposit account number 77777555/2700 administrated by GR Bank Plc.

If any of the data shown does not agree with the data in your proposal, please notify us of any required changes by post.

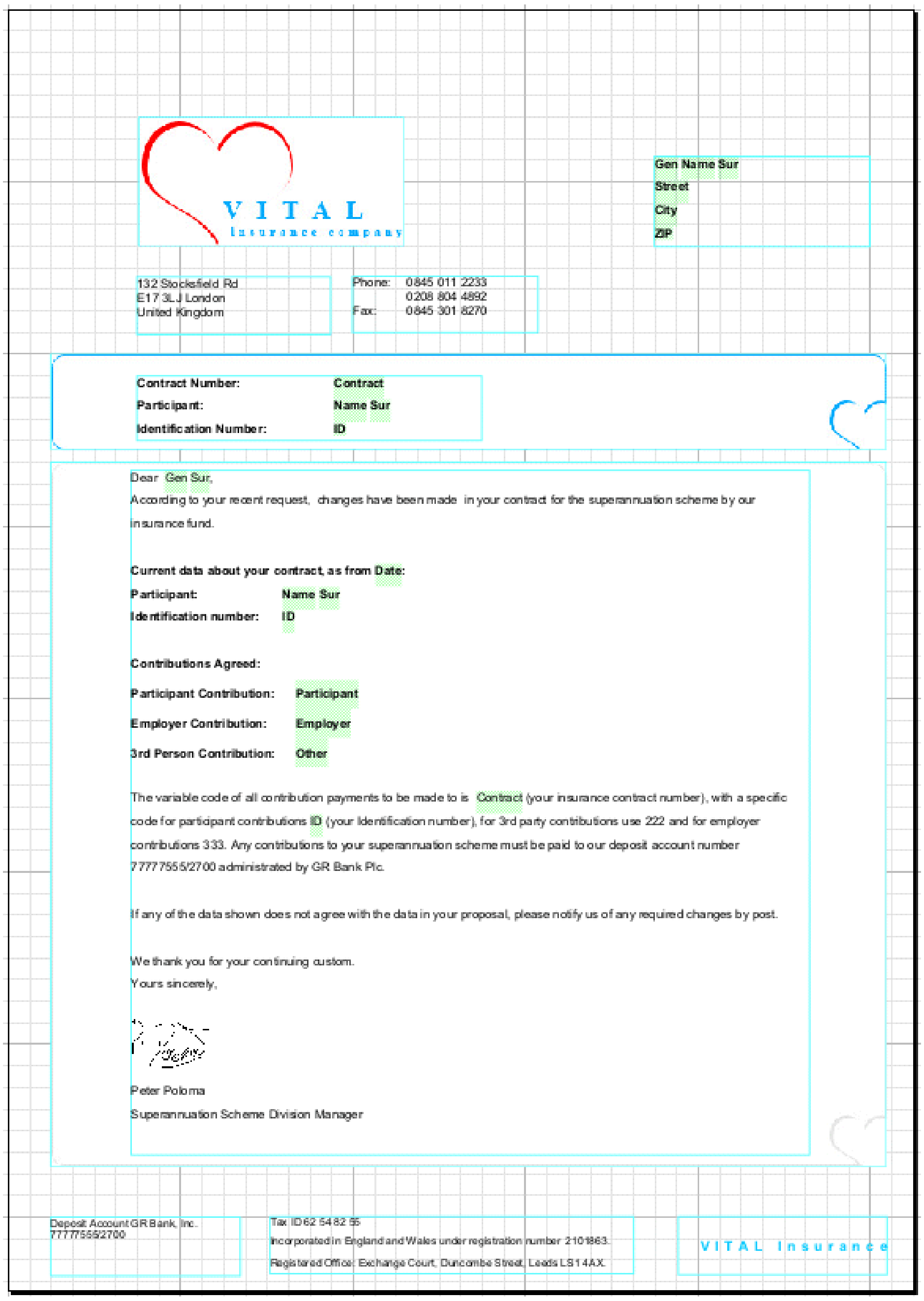
We thank you for your continuing custom.

Yours sincerely,

Peter Poloma
Superannuation Scheme Division Manager

Remember: Rename the personalized letter body flow area and the flow within it.

The design of the letter page is finished:



Gen Name Sur
Street
City
ZIP

132 Stocksfield Rd
E17 3LJ London
United Kingdom

Phone: 0845 011 2233
0208 804 4892
Fax: 0845 301 8270

Contract Number: **Contract**
Participant: **Name Sur**
Identification Number: **ID**

Dear Gen Sur,
According to your recent request, changes have been made in your contract for the superannuation scheme by our insurance fund.

Current data about your contract, as from Date:
Participant: **Name Sur**
Identification number: **ID**

Contributions Agreed:

Participant Contribution: **Participant**

Employer Contribution: **Employer**

3rd Person Contribution: **Other**

The variable code of all contribution payments to be made to is Contract (your insurance contract number), with a specific code for participant contributions ID (your Identification number), for 3rd party contributions use 222 and for employer contributions 333. Any contributions to your superannuation scheme must be paid to our deposit account number 77777555/2700 administrated by GR Bank Plc.

If any of the data shown does not agree with the data in your proposal, please notify us of any required changes by post.

We thank you for your continuing custom.
Yours sincerely,

Peter Poloma
Superannuation Scheme Division Manager

Deposit Account GR Bank, Inc.
77777555/2700

Tax ID 62 54 82 55
Incorporated in England and Wales under registration number 2101863.
Registered Office: Exchange Court, Duncombe Street, Leeds LS1 4AX.

⊕ Return to the Workflow window to add the Output module.

Click on the Workflow tab of the workflow environment .

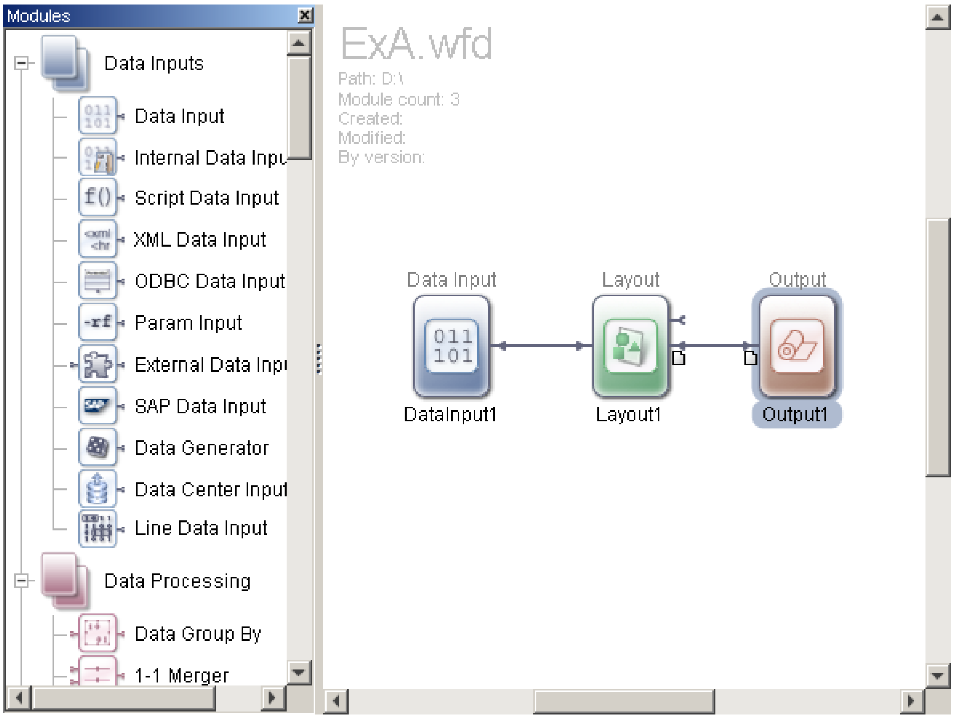
The Workflow window opens.

5.1.2.3 Output

⊕ Define the media and the device on which the designed letter should print.

Select Output from the Module Tree, drag it to the Workflow Area, drop it and connect it to the Layout module.

The Output module appears in the Layout Area and is connected as part of the workflow.



⊕ Configure the Output module.

Double-click on the Output module to open it.

Remember: The role of the Output module is to define printing media and a device profile.

In this output module we will define:

1. The media on which the workflow should be printed.
2. Parameters for the printing device by which the workflow should be printed.

Media

In our example, we only want to print one design of a page on one kind of media and, in such cases, definition of the media pages may not be required. However, we also want to define one device profile. Device profiles are part of the definition of a media page (so that every created media page can have an independent device profile). Thus, it is necessary to create a media page to be able to create a device profile.

⊕ Create a new Media Page to define the media that the letter will be printed on.

Click on the **Fill From Layout** button at the bottom of the module. A new media page using the name you entered as sheet name 1 for the page in Layout (e.g. *Letter*) will be created automatically.

Remember: When clicking on **Fill From Layout** a media page is created for each different sheet name 1 defined for a page in Layout. All media page attributes (**Sheet Name 1**, **Size**, **Width** and **Height**) will be filled according to the settings made in Layout.

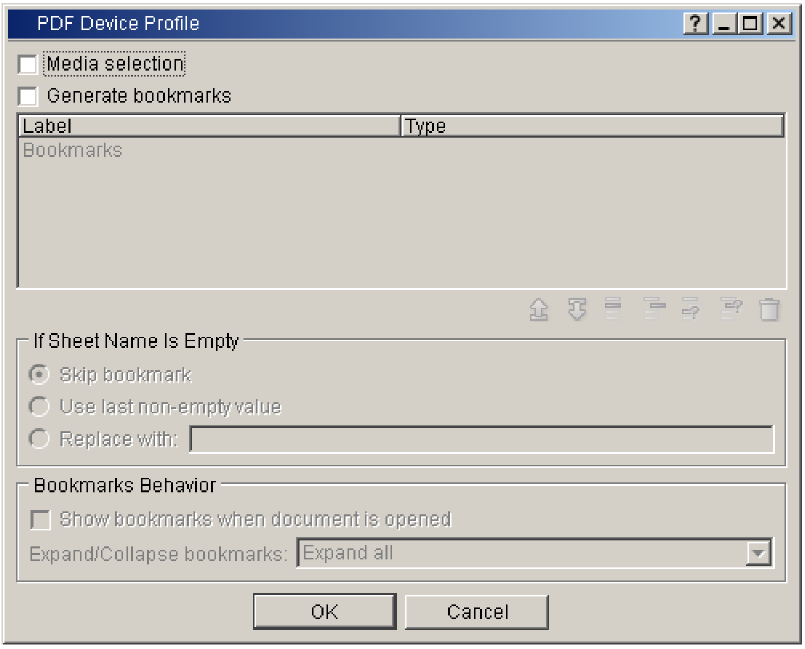
Device

⊕ Define a profile for the printing device that the workflow should be printed on.

Click on the **New** button in the **Device Profiles** field of the module.

The Create Device Profile dialog opens. Select the printer type (e.g. *PDF*) from the **Type** combo-box menu and enter a name in the **Name** edit-box, then click on **OK**.

The PDF Device Profile dialog opens:



PDF Device Profile

□ Media selection
□ Generate bookmarks

Label	Type
Bookmarks	

If Sheet Name Is Empty
● Skip bookmark
○ Use last non-empty value
○ Replace with:

Bookmarks Behavior
□ Show bookmarks when document is opened
Expand/Collapse bookmarks: Expand all

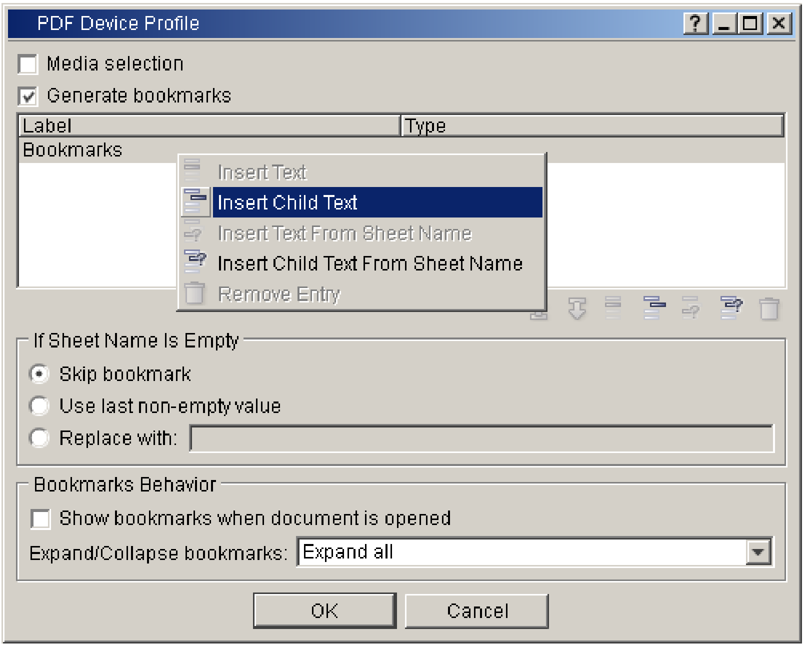
The configuration of the PDF profile allows the creation of bookmarks if required.

Remember [51]: In our Layout, we defined that sheet name 2 is assigned to the variable that reads a client's surname (e.g. Sur).

⊕ Edit the device profile so that any PDF file created when printing the workflow will have bookmarks that index it according to the second sheet name that has been set in Layout.

Check **Generate bookmarks** and right-click on *Bookmarks* in the **Label** column.

The bookmarks context menu is revealed:



PDF Device Profile

□ Media selection
☑ Generate bookmarks

Label	Type
Bookmarks	

Insert Text
Insert Child Text
Insert Text From Sheet Name
Insert Child Text From Sheet Name
Remove Entry

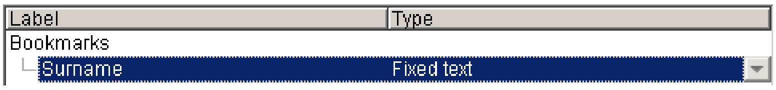
If Sheet Name Is Empty
● Skip bookmark
○ Use last non-empty value
○ Replace with:

Bookmarks Behavior
□ Show bookmarks when document is opened
Expand/Collapse bookmarks: Expand all

⊕ Add a text entry for the name of the bookmark category to identify it.

Select **Insert Child Text** and add text (e.g. *Surname*) to the **Label** column.

The bookmark category is defined:

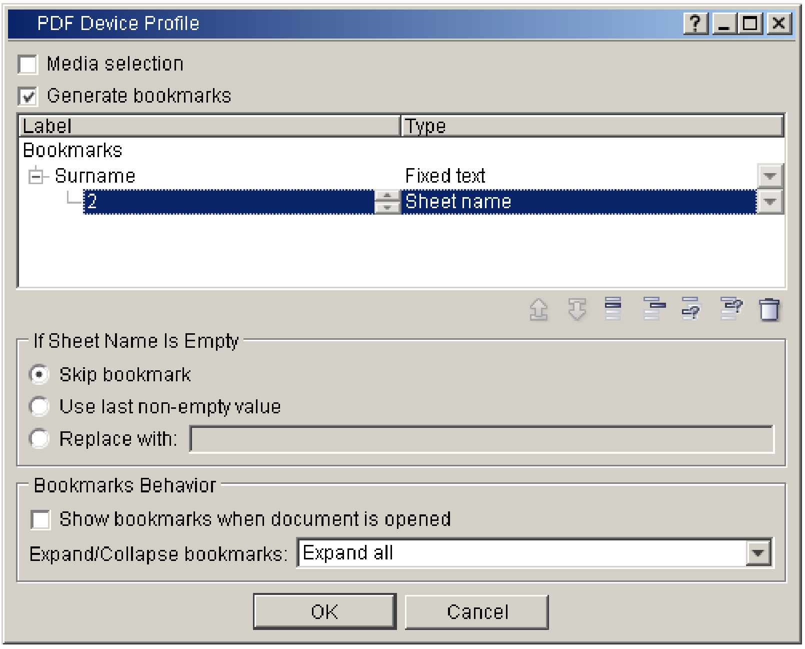


Label	Type
Bookmarks	
└ Surname	Fixed text

⊕ Assign the sheet name that will create the bookmarks and save the profile.

Select the category (e.g. *Surname*), right-click and select **Insert Child Text from Sheet Name**. In the new row that is created, enter the index number of the sheet name to the **Label** column (e.g. *2*).

The bookmarks are defined:



PDF Device Profile

☐ Media selection

☑ Generate bookmarks

Label	Type
Bookmarks	
⊟ Surname	Fixed text
└ 2	Sheet name

If Sheet Name Is Empty

● Skip bookmark

○ Use last non-empty value

○ Replace with:

Bookmarks Behavior

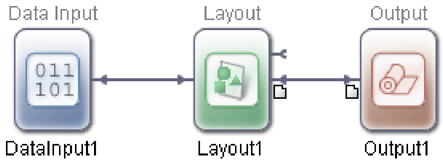
☐ Show bookmarks when document is opened

Expand/Collapse bookmarks: Expand all

Now click on **OK**. The new device profile is created and automatically selected in the **Profile** combo-box of the Output configuration dialog.

To return to the Workflow window click on **OK** to close the Output configuration dialog.

5.1.2.4 Workflow Completion

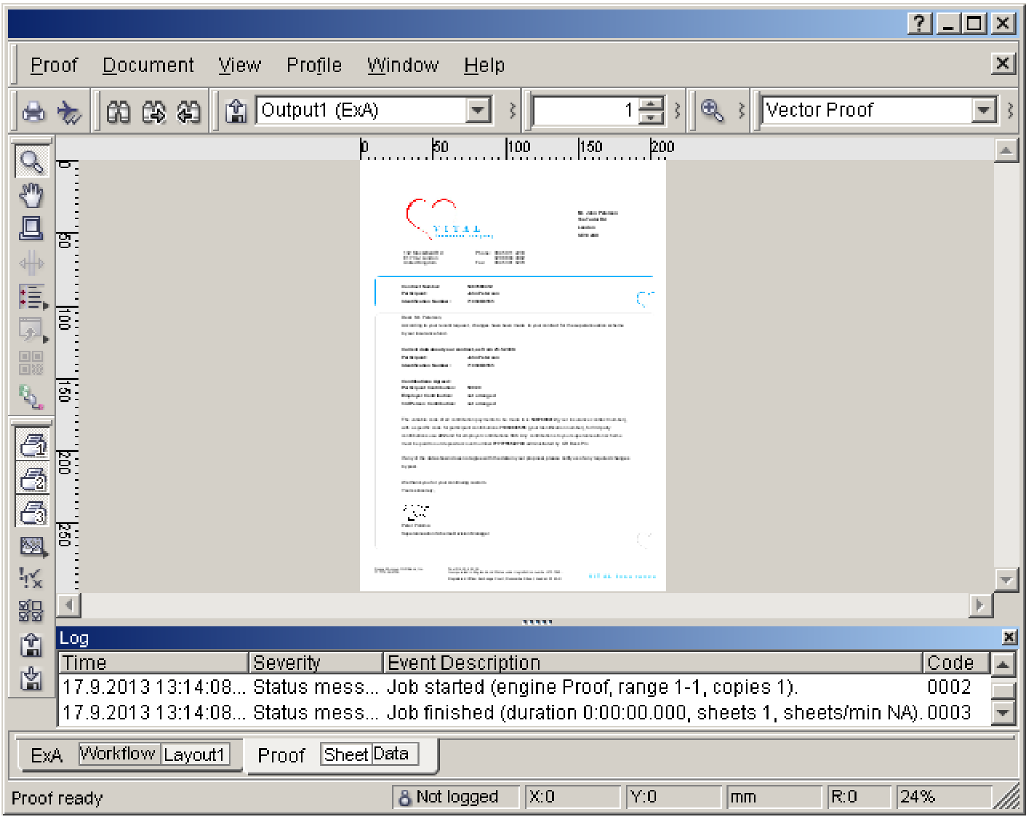


⊕ Save the workflow and check that the design would print how it is expected to.

Go to the menu **File | Save** and then click on the **Proof** icon on the Workflow toolbar or press <**F6**>.

The Proof environment opens.

5.1.3 Proof



```
Time                Severity       Event Description                                                 Code
17.9.2013 13:14:08... Status mess... Job started (engine Proof, range 1-1, copies 1).                 0002
17.9.2013 13:14:08... Status mess... Job finished (duration 0:00:00.000, sheets 1, sheets/min NA). 0003
```

In this example, we have determined that a one page letter will be printed for each record in the data file (i.e. each client), therefore there should be as many pages as there are clients in the data file.

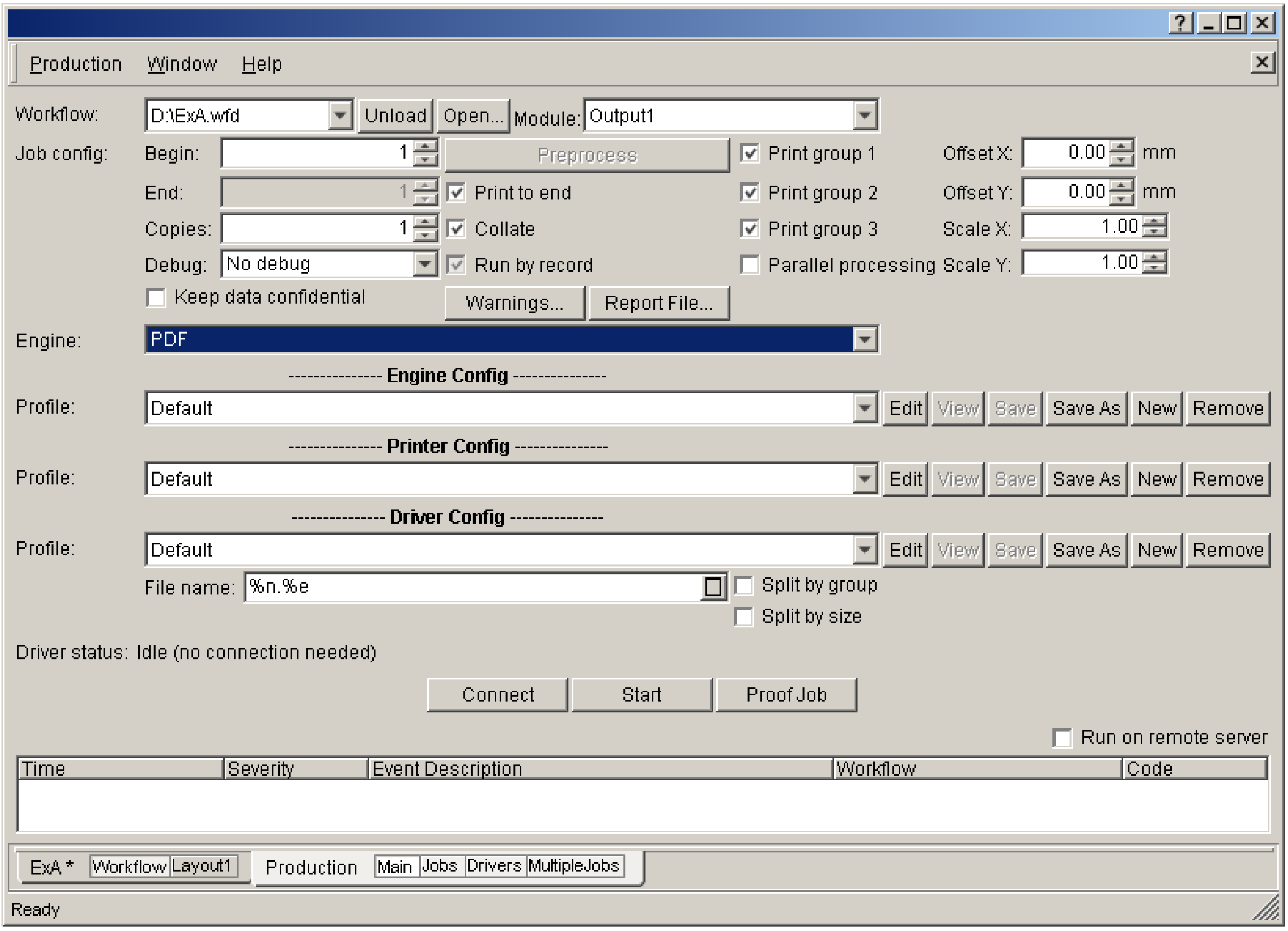
Remember: You can scroll through all of the proofed pages using the **Pages** toolbar.

⊕ Load the approved workflow to Production so that the required PDF output can be generated.

Return to the Workflow window using the tab ExA * Workflow Layout1 and click on the **Production** icon on the Workflow toolbar or press <**F5**>.

The Production environment opens with the workflow loaded to it:

5.1.4 Production



Workflow: D:\ExA.wfd
Module: Output1
Job config: Begin: 1
End: 1
Copies: 1
Debug: No debug
☑ Print to end
☑ Collate
☑ Run by record
☑ Print group 1
☑ Print group 2
☑ Print group 3
□ Parallel processing
Offset X: 0.00 mm
Offset Y: 0.00 mm
Scale X: 1.00
Scale Y: 1.00
□ Keep data confidential
Engine: PDF
Engine Config Profile: Default
Printer Config Profile: Default
Driver Config Profile: Default
File name: %n.%e
□ Split by group
□ Split by size
Driver status: Idle (no connection needed)
□ Run on remote server

Time	Severity	Event Description	Workflow	Code

Ready

⊕ Select the type of page description language you would like the print job to use.

Make sure *PDF* is selected in the **Engine** combo-box.

⊕ Create a printer engine profile to be configured to the output requirements.

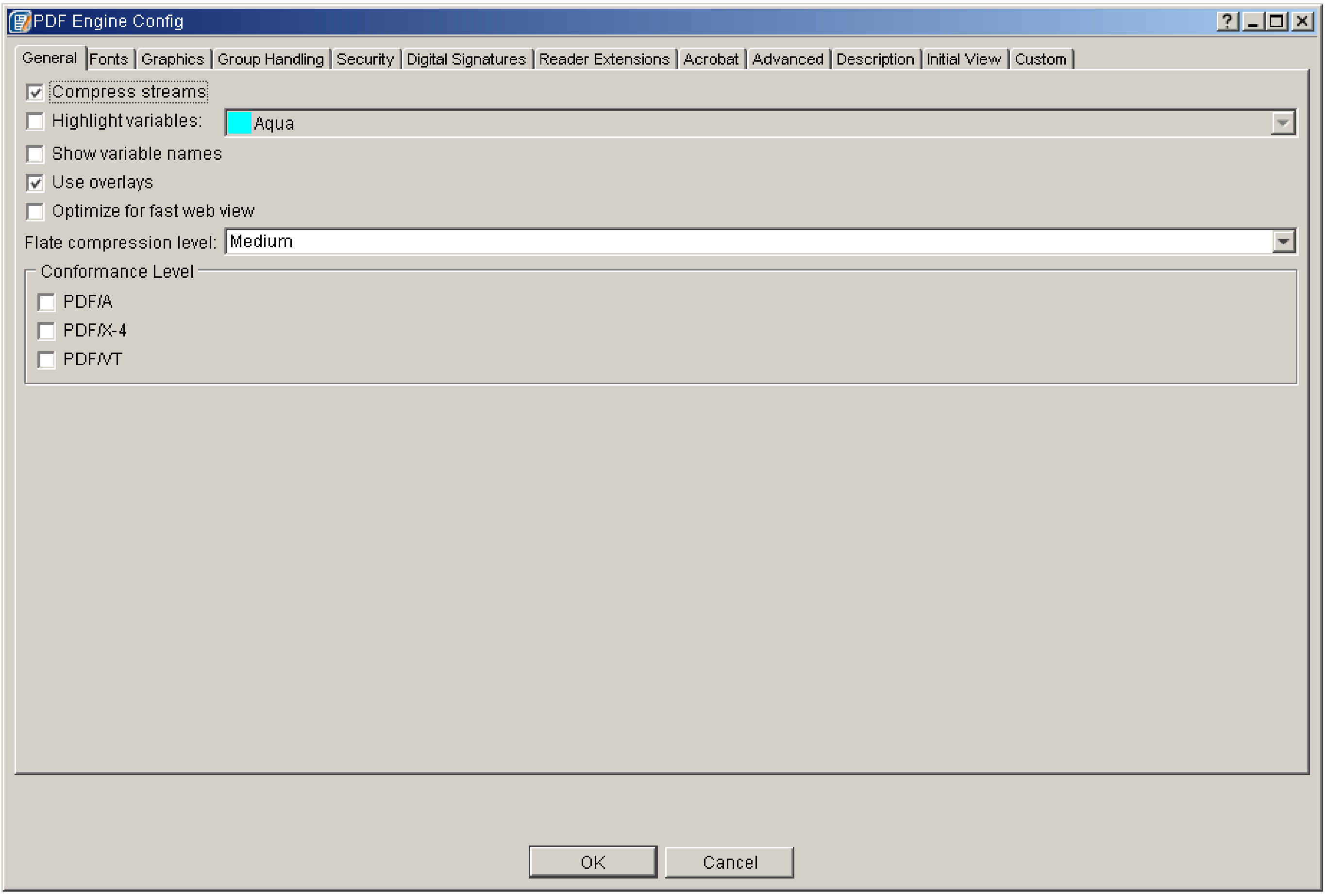
Click on the **New** button that is in the Engine Config field and enter a name for the profile in the **Name** edit-box of the Create Engine Profile dialog that opens. Then click on **OK**.

The engine profile is created and can be selected in the **Profile** combo-box whenever the same engine type (e.g. PDF) is selected in the **Engine** combo-box.

⊕ If needed, edit the engine configuration to fit the output requirements.

Click on the **Edit** button.

The Engine Config dialog opens:


PDF Engine Config
General | Fonts | Graphics | Group Handling | Security | Digital Signatures | Reader Extensions | Acrobat | Advanced | Description | Initial View | Custom
OK
Cancel

☑ Compress streams
□ Highlight variables: Aqua
□ Show variable names
☑ Use overlays
□ Optimize for fast web view
Flate compression level: Medium
Conformance Level
□ PDF/A
□ PDF/X-4
□ PDF/VT

In our example, we do not want to make any adjustments to the default settings.

Remember: The Engine Config dialog allows the editing of the file processing settings for an engine type.

⊕ Confirm the output format settings.

Click on **OK**.

The settings are saved and the dialog is dismissed.

⊕ Create a printer driver profile to be configured to the output requirements.

Click on the **New** button that is in the Driver Config field of the Production window and enter a name for the profile in the **Name** edit-box of the Create Driver Profile dialog that opens. Then click on **OK**.

The driver profile is created and can be selected in the **Profile** combo-box whenever the same engine type (e.g. PDF) is selected in the **Engine** combo-box.

Remember: There are substitution characters available in order to name the produced file automatically (e.g. %n is the name of workflow and %e is the default extension of the engine).

⊕ Give the file to be printed as a PDF document a name and specify a directory where it should be saved.

Enter a path to a location and a name in the **File Name** edit-box (e.g. *C:\Documents\%n.%e* will create the file ExA.pdf in the Documents directory).

The job will be given a name when printed.

⊕ Print the workflow.

Click on **Start**.

The Jobs window opens.

⊕ Collect the finished job from the location that was defined in the **File Name** combo-box of the Driver Config.

When the job's log stops spooling and if no errors occur, go to the directory (e.g. C:\Documents) and open the file (e.g. ExA.pdf).

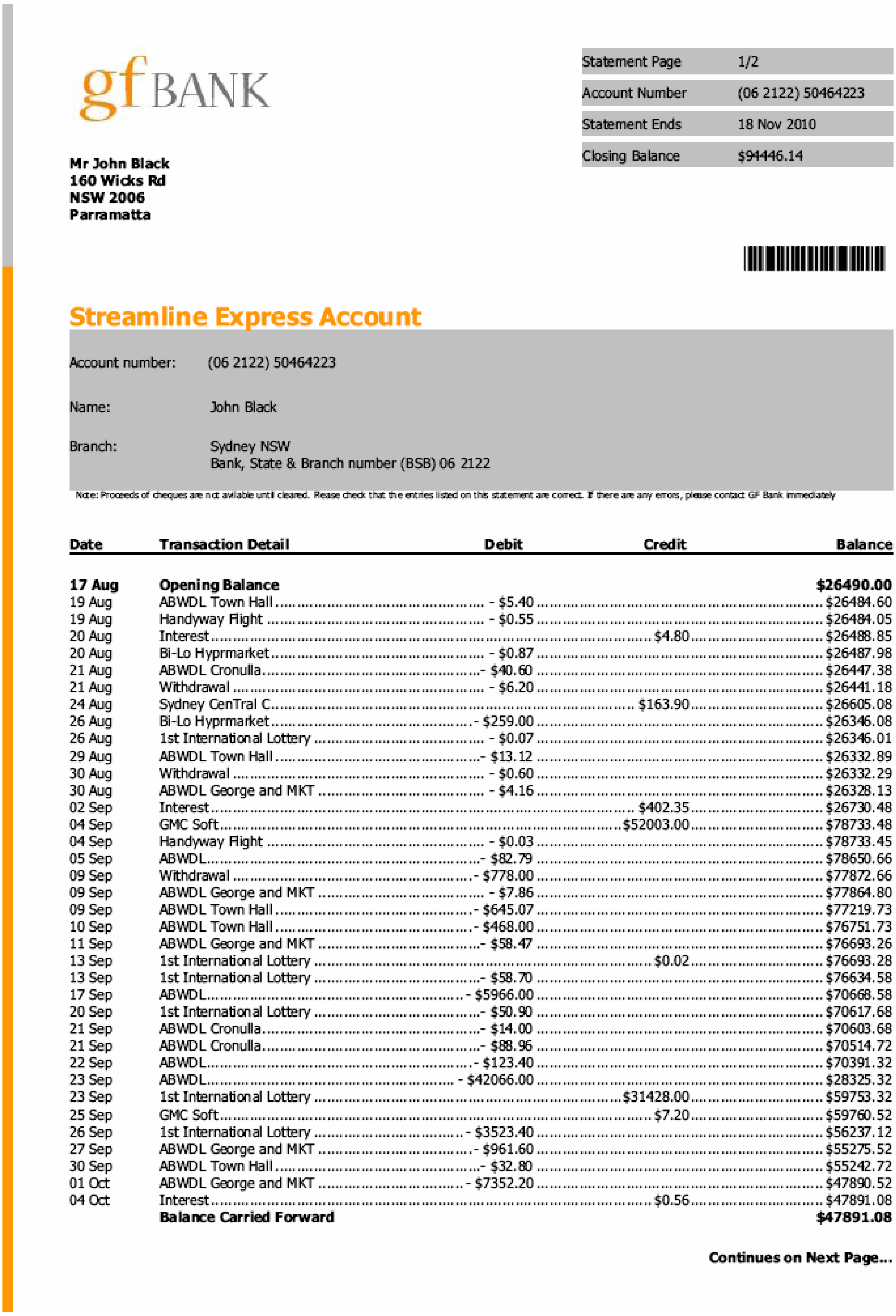
Another software application (a PDF reader) must be used to view the file itself and the bookmarks [52] it contains. The file can be also sent to a printer.

5.2 Example B – Account Statement

Another common use for Inspire Designer is in the banking industry. Banks have many clients, each of whom have different accounts with differing daily transactions.

As a scenario for this example, all of the clients' personal details are stored in one data file and details about transactions made by all of a bank's clients are listed in another data format. In addition, there is a separate data file where details of each branch of the bank are held.

For monthly account statements, a standard statement template is designed to cover the issue and (because each client has different personal details, branch of bank and transactions) all of these personal details need to be collected together and written on the single statement for an individual client. Further to that, the design of the statement must be able to cater for differing amounts of transaction data, i.e. the fields of similar data (the transactions) being repeated an unknown amount of times.



gf BANK

Statement Page	1/2
Account Number	(06 2122) 50464223
Statement Ends	18 Nov 2010
Closing Balance	$94446.14

Mr John Black
160 Wicks Rd
NSW 2006
Parramatta

Streamline Express Account

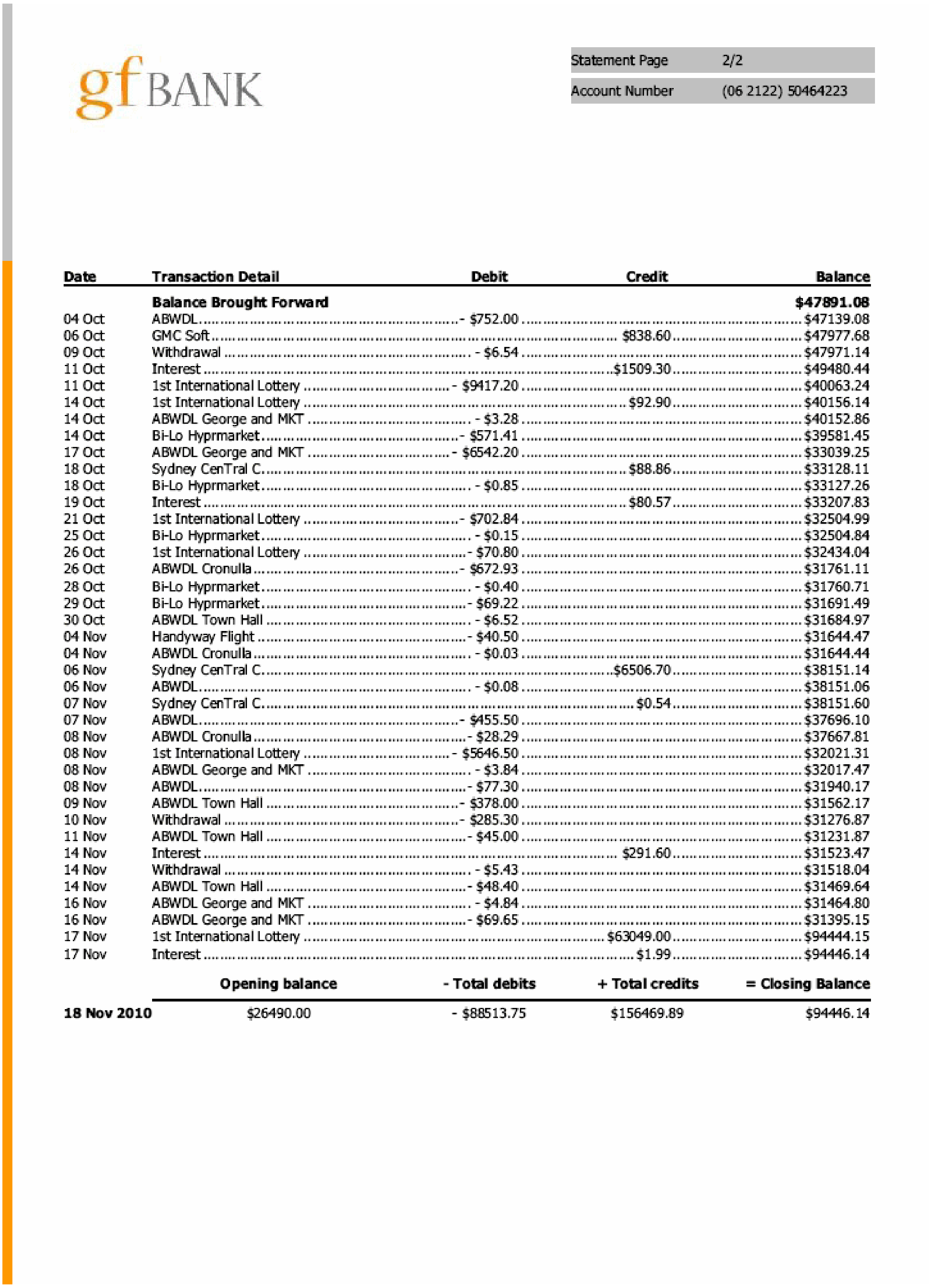
Account number:	(06 2122) 50464223
Name:	John Black
Branch:	Sydney NSW Bank, State & Branch number (BSB) 06 2122

Note: Proceeds of cheques are not avilable until cleared. Please check that the entries listed on this statement are correct. If there are any errors, please contact GF Bank immediately

Date	Transaction Detail	Debit	Credit	Balance
17 Aug	**Opening Balance**			**$26490.00**
19 Aug	ABWDL Town Hall	- $5.40		$26484.60
19 Aug	Handyway Flight	- $0.55		$26484.05
20 Aug	Interest		$4.80	$26488.85
20 Aug	Bi-Lo Hyprmarket	- $0.87		$26487.98
21 Aug	ABWDL Cronulla	- $40.60		$26447.38
21 Aug	Withdrawal	- $6.20		$26441.18
24 Aug	Sydney CenTral C.		$163.90	$26605.08
26 Aug	Bi-Lo Hyprmarket	- $259.00		$26346.08
26 Aug	1st International Lottery	- $0.07		$26346.01
29 Aug	ABWDL Town Hall	- $13.12		$26332.89
30 Aug	Withdrawal	- $0.60		$26332.29
30 Aug	ABWDL George and MKT	- $4.16		$26328.13
02 Sep	Interest		$402.35	$26730.48
04 Sep	GMC Soft		$52003.00	$78733.48
04 Sep	Handyway Flight	- $0.03		$78733.45
05 Sep	ABWDL	- $82.79		$78650.66
09 Sep	Withdrawal	- $778.00		$77872.66
09 Sep	ABWDL George and MKT	- $7.86		$77864.80
09 Sep	ABWDL Town Hall	- $645.07		$77219.73
10 Sep	ABWDL Town Hall	- $468.00		$76751.73
11 Sep	ABWDL George and MKT	- $58.47		$76693.26
13 Sep	1st International Lottery		$0.02	$76693.28
13 Sep	1st International Lottery	- $58.70		$76634.58
17 Sep	ABWDL	- $5966.00		$70668.58
20 Sep	1st International Lottery	- $50.90		$70617.68
21 Sep	ABWDL Cronulla	- $14.00		$70603.68
21 Sep	ABWDL Cronulla	- $88.96		$70514.72
22 Sep	ABWDL	- $123.40		$70391.32
23 Sep	ABWDL	- $42066.00		$28325.32
23 Sep	1st International Lottery		$31428.00	$59753.32
25 Sep	GMC Soft		$7.20	$59760.52
26 Sep	1st International Lottery	- $3523.40		$56237.12
27 Sep	ABWDL George and MKT	- $961.60		$55275.52
30 Sep	ABWDL Town Hall	- $32.80		$55242.72
01 Oct	ABWDL George and MKT	- $7352.20		$47890.52
04 Oct	Interest		$0.56	$47891.08
	Balance Carried Forward			**$47891.08**

Continues on Next Page...

Figure 5.10 An Example Design for the Pages of the Account Statement - Page 1



gfBANK

Statement Page	2/2
Account Number	(06 2122) 50464223

Date	Transaction Detail	Debit	Credit	Balance
	Balance Brought Forward			**$47891.08**
04 Oct	ABWDL	- $752.00		$47139.08
06 Oct	GMC Soft		$838.60	$47977.68
09 Oct	Withdrawal	- $6.54		$47971.14
11 Oct	Interest		$1509.30	$49480.44
11 Oct	1st International Lottery	- $9417.20		$40063.24
14 Oct	1st International Lottery		$92.90	$40156.14
14 Oct	ABWDL George and MKT	- $3.28		$40152.86
14 Oct	Bi-Lo Hyprmarket	- $571.41		$39581.45
17 Oct	ABWDL George and MKT	- $6542.20		$33039.25
18 Oct	Sydney CenTral C.		$88.86	$33128.11
18 Oct	Bi-Lo Hyprmarket	- $0.85		$33127.26
19 Oct	Interest		$80.57	$33207.83
21 Oct	1st International Lottery	- $702.84		$32504.99
25 Oct	Bi-Lo Hyprmarket	- $0.15		$32504.84
26 Oct	1st International Lottery	- $70.80		$32434.04
26 Oct	ABWDL Cronulla	- $672.93		$31761.11
28 Oct	Bi-Lo Hyprmarket	- $0.40		$31760.71
29 Oct	Bi-Lo Hyprmarket	- $69.22		$31691.49
30 Oct	ABWDL Town Hall	- $6.52		$31684.97
04 Nov	Handyway Flight	- $40.50		$31644.47
04 Nov	ABWDL Cronulla	- $0.03		$31644.44
06 Nov	Sydney CenTral C.		$6506.70	$38151.14
06 Nov	ABWDL	- $0.08		$38151.06
07 Nov	Sydney CenTral C.		$0.54	$38151.60
07 Nov	ABWDL	- $455.50		$37696.10
08 Nov	ABWDL Cronulla	- $28.29		$37667.81
08 Nov	1st International Lottery	- $5646.50		$32021.31
08 Nov	ABWDL George and MKT	- $3.84		$32017.47
08 Nov	ABWDL	- $77.30		$31940.17
09 Nov	ABWDL Town Hall	- $378.00		$31562.17
10 Nov	Withdrawal	- $285.30		$31276.87
11 Nov	ABWDL Town Hall	- $45.00		$31231.87
14 Nov	Interest		$291.60	$31523.47
14 Nov	Withdrawal	- $5.43		$31518.04
14 Nov	ABWDL Town Hall	- $48.40		$31469.64
16 Nov	ABWDL George and MKT	- $4.84		$31464.80
16 Nov	ABWDL George and MKT	- $69.65		$31395.15
17 Nov	1st International Lottery		$63049.00	$94444.15
17 Nov	Interest		$1.99	$94446.14

	Opening balance	- Total debits	+ Total credits	= Closing Balance
18 Nov 2010	$26490.00	- $88513.75	$156469.89	$94446.14

Figure 5.11 An Example Design for the Pages of the Account Statement - Page 2

Introducing the features of the design of the account statement:

1. Two Pages (design for first page and a continuation page)
2. Logo and Template Shapes (static objects)
3. Barcode (a variable object)

4. Client Address Area (consists of variable text)

5. Statement Details Area (consists of static and variable text)

6. Account Details Area (consists of static and variable text)

7. Account Statement Body (consists of static and variable text for the first page, overflowing to the continuation page if required)

8. Correctly Formatted Data (dates and currency values should be formatted as part of the design process)

Normal practice would be to have template details (2) pre-printed onto two differently designed media pages and then control the content (which includes personalized details (3-7) and page selection properties) with the workflow. However, for the purpose of this example it is instructive to add all of the features as if to print on a blank media.

The function of the workflow we create will be to:

1. Read from the CSV file some client details, including their account number and which branch of the bank they belong to.

2. Match the bank branch to the corresponding bank branch details in a TXT file and read those details.

3. Match the clients to all of the transactions that they made from those listed in the XML transactions list and read those details.

4. Print the complete set of matched data to a design of a multiple page letter (a single first page and a continuation page that can be repeated).

In order to perform that function, the workflow will be composed of ten modules:

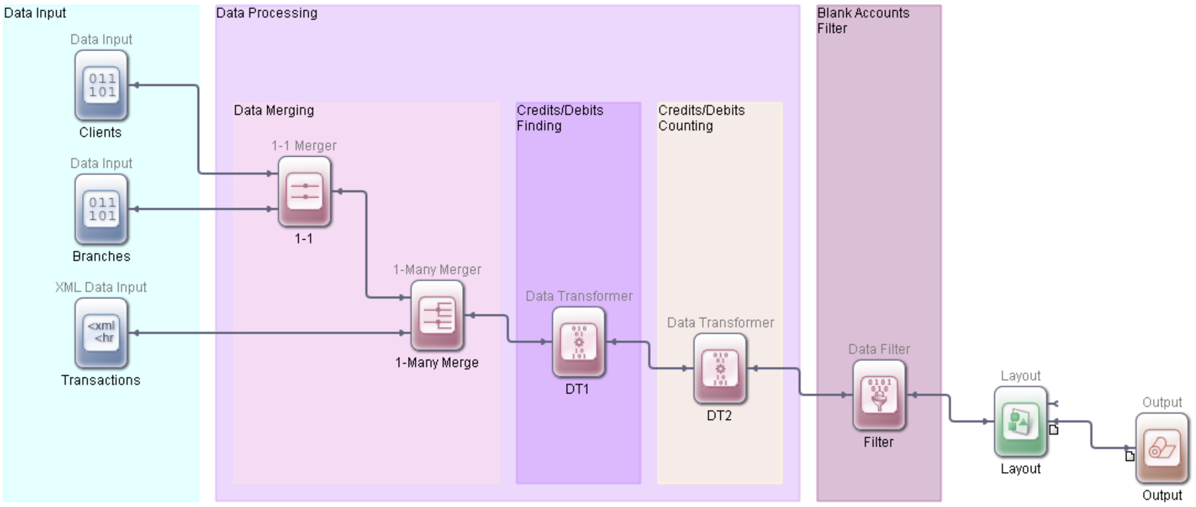


Figure 5.12 The Modules of Example B

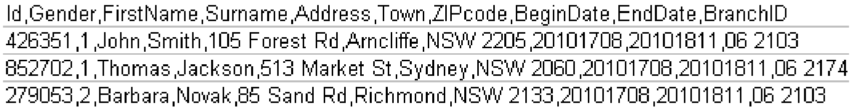
The labels placed in the background of the modules are intended to make it easier to keep an overview when complicated workflows consisting of many modules are used. To create labels, select **Workflow | Label Creating**, go to the Workflow area, press the left mouse button and drag the mouse to create a label of the desired size. Right-click on the label and select **Edit Label** from the context menu to open the Edit Label dialog where you can enter the text displayed inside of the label, its alignment and the color of the label. Using the context menu and the options **Move Label Backward** and **Move Label Forward** available there, you can determine the z-order of labels which is useful when labels overlap.

5.2.1 Requirements

To complete Example B, some external files are required so that they can be read or inserted into the workflow and become objects that appear in the account statement:

- A 'client detail' data file (a CSV file) that contains each client's personal details.

  To compile the example, our CSV file contains the following fields entered in String data-type: Id, Gender, FirstName, Surname, Address, Town, ZIPCode, BeginDate (the account statement's start date), EndDate (the account statement's end date), and BranchID (a code for the client's branch of the bank):

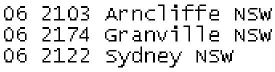
  

  ```
  Id,Gender,FirstName,Surname,Address,Town,ZIPcode,BeginDate,EndDate,BranchID
  426351,1,John,Smith,105 Forest Rd,Arncliffe,NSW 2205,20101708,20101811,06 2103
  852702,1,Thomas,Jackson,513 Market St,Sydney,NSW 2060,20101708,20101811,06 2174
  279053,2,Barbara,Novak,85 Sand Rd,Richmond,NSW 2133,20101708,20101811,06 2103
  ```

  The first three records of the CSV file that was used when compiling this example

- A 'bank branch' data file (a 'fixed length fields' TXT file) with corresponding addresses of each bank from the 'client detail' data file.

  To compile the example, our TXT file contains the following fields entered in String data-type: BranchCode (a code for each branch of the bank, which always corresponds to a number that can be found in the BranchID field of the 'client detail' data file) and Location (the town in which the branch is in):

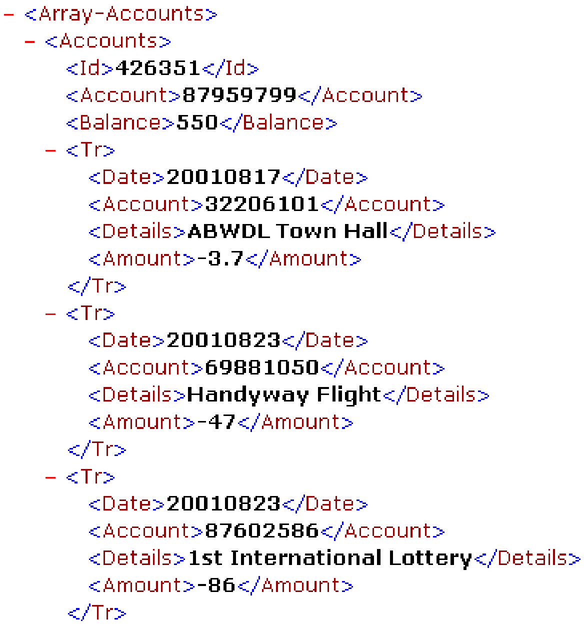
  

  ```
  06 2103 Arncliffe NSW
  06 2174 Granville NSW
  06 2122 Sydney NSW
  ```

  The first three records of the TXT file that was used when compiling this example

- A 'transaction' file (an XML file) with many transactions for each of the clients of the 'client detail' data file. This considers that it is possible for one client to have two accounts but that not all of those accounts must have transactions on them.

  To compile the example, our XML file contained the following elements: Id (identification numbers that corresponds to those in the ID field of the 'client detail' data file), Account (number of accounts which the transactions were made on), Balance (the account balance at the start of the statement) and Transactions which is an element of array data-type and contains the further elements Date (the date of each transaction), Account (the transaction's receiving or sending account), Details (information about that receiving/sending account), Amount (the transaction value – no currency symbol included):



```
- <Array-Accounts>
  - <Accounts>
      <Id>426351</Id>
      <Account>87959799</Account>
      <Balance>550</Balance>
    - <Tr>
        <Date>20010817</Date>
        <Account>32206101</Account>
        <Details>ABWDL Town Hall</Details>
        <Amount>-3.7</Amount>
      </Tr>
    - <Tr>
        <Date>20010823</Date>
        <Account>69881050</Account>
        <Details>Handyway Flight</Details>
        <Amount>-47</Amount>
      </Tr>
    - <Tr>
        <Date>20010823</Date>
        <Account>87602586</Account>
        <Details>1st International Lottery</Details>
        <Amount>-86</Amount>
      </Tr>
```

The first element of the XML file that was used when compiling this example. Showing only the first three transaction elements on the account.

- An image file to be used for the logo at the top of the statement.

  To compile our example, we will insert the bank logo as a JPG file:

5.2.2 Workflow

⊕ Open a new workflow.

Start Inspire Designer and select **New Workflow** from the Startup dialog.

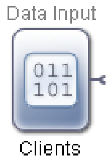
The Workflow window opens.

Remember: Save the new workflow with a new name in an easy-to-find location (e.g. ExB.wfd in the same directory as the images and data file that you will use).

5.2.2.1 Data Input 1

⊕ Add a Data Input module that can read the client's personal details from the CSV file.

Drag a Data Input module to the Workflow Area, drop it there and rename it (e.g. to "Clients"):



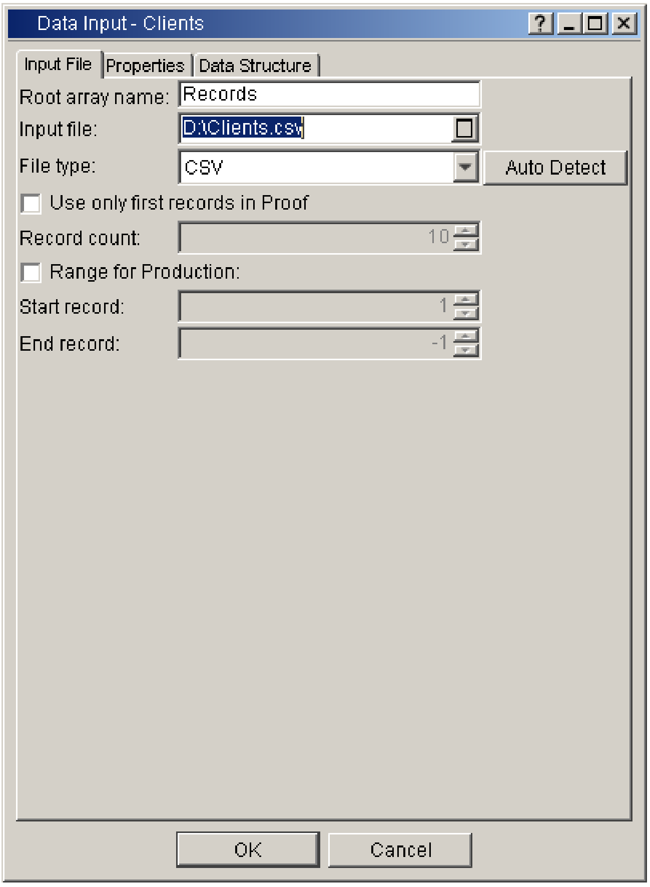
⊕ Configure the Data Input module so it can read the format of the data file.

Double-click on the module.

The module opens.

⊕ Enter the path to the CSV file.

In the **Input file** edit-box, browse □ for the CSV file (e.g. Clients.csv) and click on **Open**. The full path to the file is displayed.



Root array name: Records

Input file: D:\Clients.csv

File type: CSV

□ Use only first records in Proof

Record count: 10

□ Range for Production:

Start record: 1

End record: -1

The attribute **File type** should be checked to see if it matches the data file.

⊕ Check the attributes to see that they match the format of the data file.

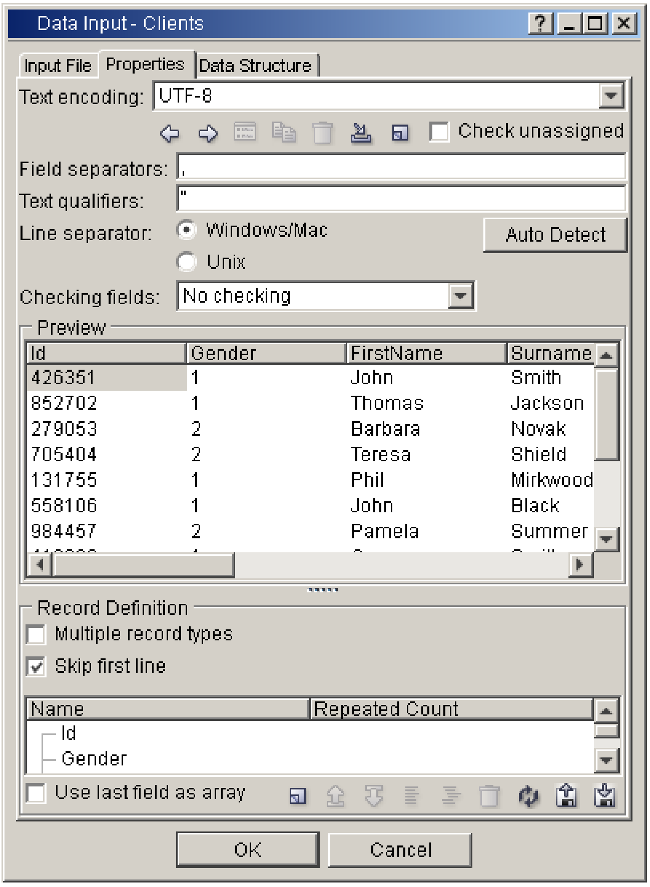
Click on the **Auto Detect** button in order to correctly fill these attributes.

All attributes on the Input File tab are set.

⊕ Set CSV file type specific parameters.

Go to the Properties tab.

The attributes of the CSV file are displayed:



Text encoding: UTF-8

□ Check unassigned

Field separators: ,

Text qualifiers: "

Line separator: ● Windows/Mac ○ Unix

Checking fields: No checking

Id	Gender	FirstName	Surname
426351	1	John	Smith
852702	1	Thomas	Jackson
279053	2	Barbara	Novak
705404	2	Teresa	Shield
131755	1	Phil	Mirkwood
558106	1	John	Black
984457	2	Pamela	Summer

□ Multiple record types

☑ Skip first line

Name	Repeated Count
Id	
Gender	

□ Use last field as array

The Properties tab shows a **Preview** of the data that it is read from the data file, making it possible to review the file for obvious errors in format. The **Text encoding** combo-box selects the codec to be used for the text. Check that the selected codec recognizes all of the characters from your data file.

⊕ Compare the attributes on the Properties tab with the structure of the data file, to check that they are correct.

Check that the attributes **Field separator**, **Text qualifiers** and **Line separator** match the actual format of the data file that is used. If not or you are not sure how the file is formatted, click on **Auto Detect** button to correctly fill these attributes.

The data file fields show correctly in the **Preview** area.

⊕ Confirm the configuration and return to the Workflow window to add the next module.

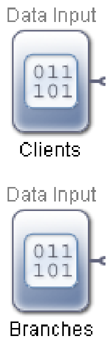
Click on **OK**.

The Data Input module closes.

5.2.2.2 Data Input 2

⊕ Add a second Data Input module to read the bank branch details from the 'fixed length fields' data file.

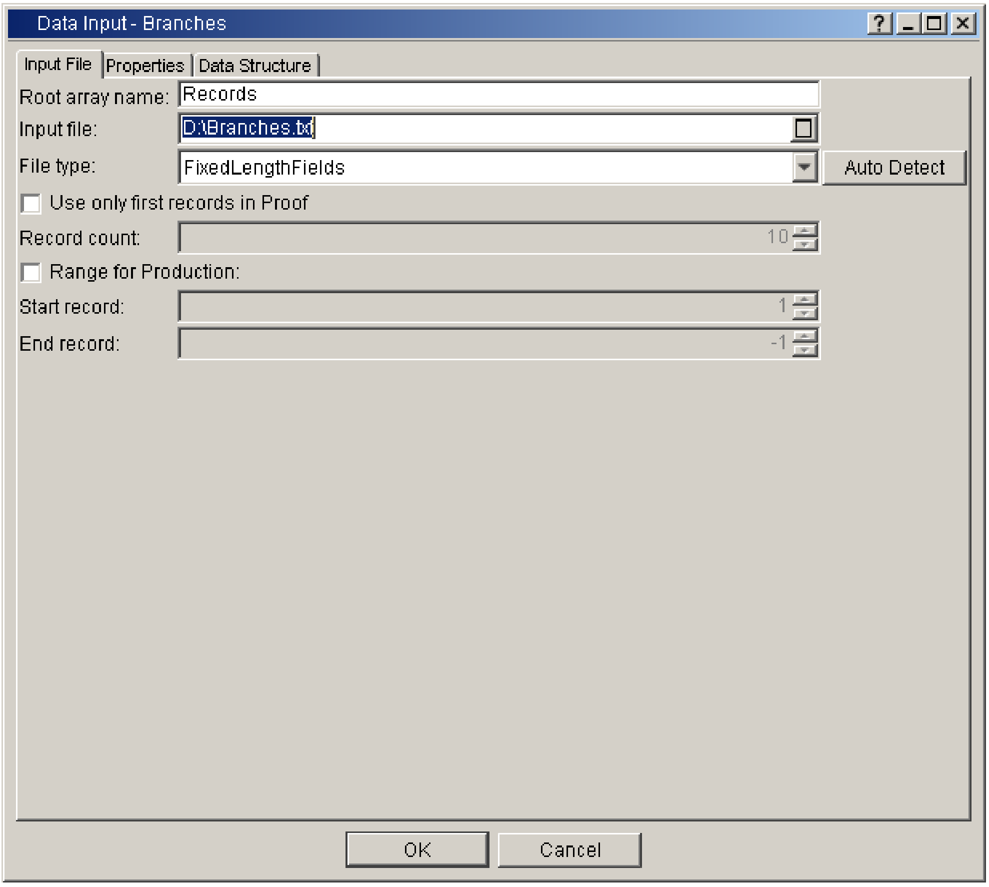
Drag another Data Input module to the Workflow Area, drop it there and rename it (e.g. to "Branches"):



⊕ Configure the second Data Input module so that it can read the TXT file which is formatted as a 'fixed length fields' data file:

Double-click on the module to open it.

In the **Input file** edit-box, browse □ for the TXT file (e.g. Branches.txt) and click on **Open**. Then select *FixedLengthFields* from the **File type** combo-box menu.

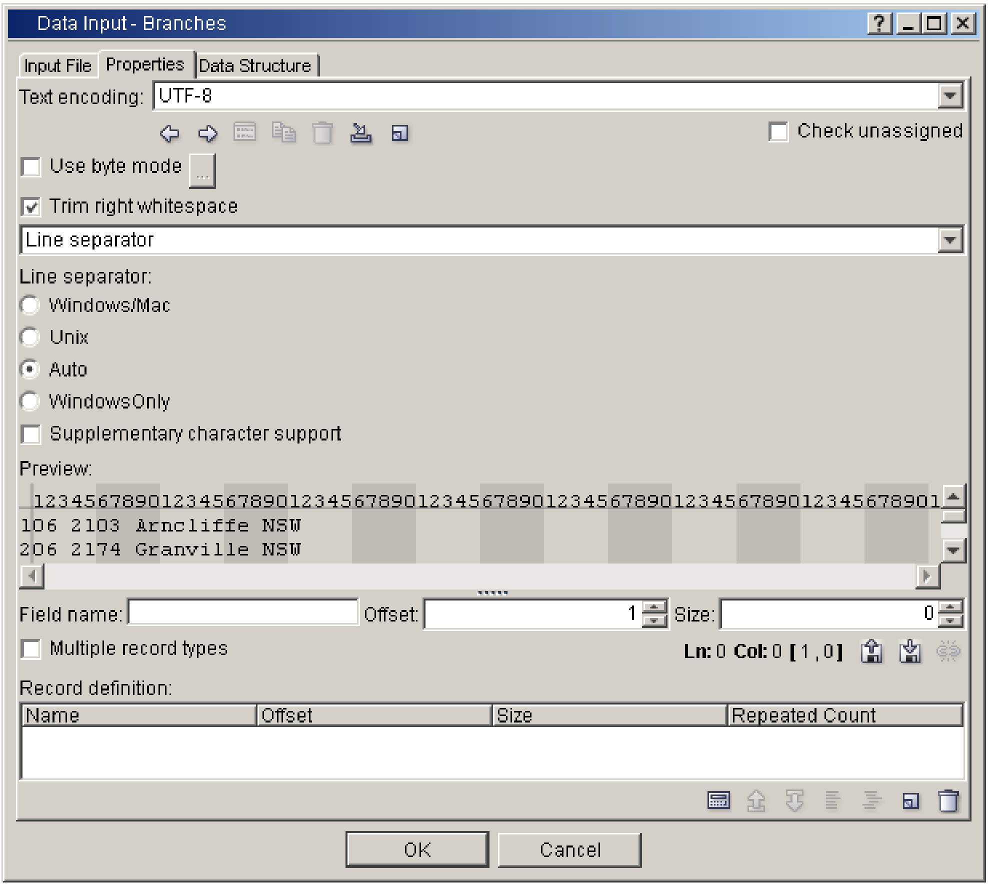


Root array name: Records
Input file: D:\Branches.txt
File type: FixedLengthFields
□ Use only first records in Proof
Record count: 10
□ Range for Production:
Start record: 1
End record: -1

⊕ Set 'fixed length fields' specific parameters.

Go to the Properties tab.

The attributes of the file are displayed:



Data Input - Branches

Input File | Properties | Data Structure

Text encoding: UTF-8

☐ Check unassigned

☐ Use byte mode

☑ Trim right whitespace

Line separator

Line separator:

○ Windows/Mac

○ Unix

● Auto

○ WindowsOnly

☐ Supplementary character support

Preview:

```
12345678901234567890123456789012345678901234567890123456789012345678901
106 2103 Arncliffe NSW
206 2174 Granville NSW
```

Field name: Offset: 1 Size: 0

☐ Multiple record types

Ln: 0 Col: 0 [1 , 0]

Record definition:

Name	Offset	Size	Repeated Count

OK Cancel

The Properties tab shows a preview of the data that it is read from the data file, making it possible to review the file for obvious errors in format. The **Text encoding** combo-box selects the codec to be used for the text. Be careful that the encoding that you select recognizes all of the characters from your data file.

⊕ Compare the attributes on the Properties tab with the structure of the data file, to check that they are correct.

Check that **Line separator** matches the actual format of the data file that is used, i.e. *Windows/Mac* or *Unix*. If your data file does not have the same parameters the records will not show correctly in the preview area.

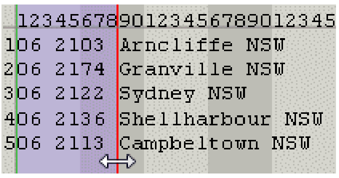
The properties set match the format of the data file.

The nature of the 'fixed length fields' file is that the fields of each record are left undefined. The fields have to be identified in order for the workflow to be able to read the records as an array of data.

⊕ Define the first field in the data file.

Click on the **Create New** icon at the bottom of the dialog. Select the intended content of the field (i.e. from position where the field content starts, to the position where the content of the next field starts) in the **Preview** area.

The contents of the first field are highlighted:



```
1234567890123456789012345
106 2103 Arncliffe NSW
206 2174 Granville NSW
306 2122 Sydney NSW
406 2136 Shellharbour NSW
506 2113 Campbeltown NSW
```

⊕ Assign a name to the first field.

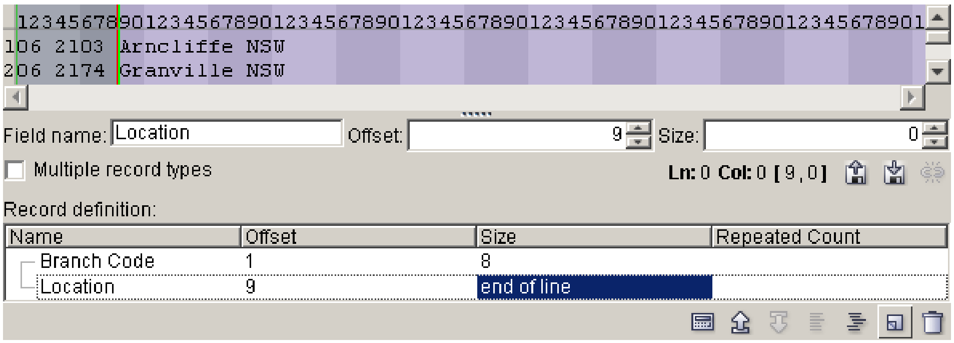
Give the field a name in the **Field name** edit-box that reflects the nature of its content (e.g. Branch Code).

The field is defined.

⊕ Define the other fields.

Repeat the above two steps, this time highlighting the second field and naming it according to its own function (e.g. Location).

The fields are defined:



```
 12345678901234567890123456789012345678901234567890123456789012345678901
106 2103 Arncliffe NSW
206 2174 Granville NSW
```

Field name: Location Offset: 9 Size: 0

□ Multiple record types Ln: 0 Col: 0 [9,0]

Record definition:

Name	Offset	Size	Repeated Count
Branch Code	1	8	
Location	9	end of line	

⊕ Confirm the configuration and return to the Workflow window to add the next module.

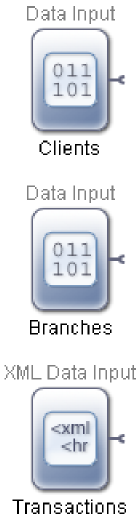
Click on **OK**.

The Data Input module closes.

5.2.2.3 XML Data Input

⊕ Add an XML Data Input module to read the clients' transaction details from the data file that is in XML format.

Drag an XML Data Input module to the Workflow Area, drop it there and rename it (e.g. to "Transactions"):



⊕ Configure the XML Data Input module so it can read the format of the data file:

Double-click on the module.

The module opens.

⊕ Enter the path to the XML file.

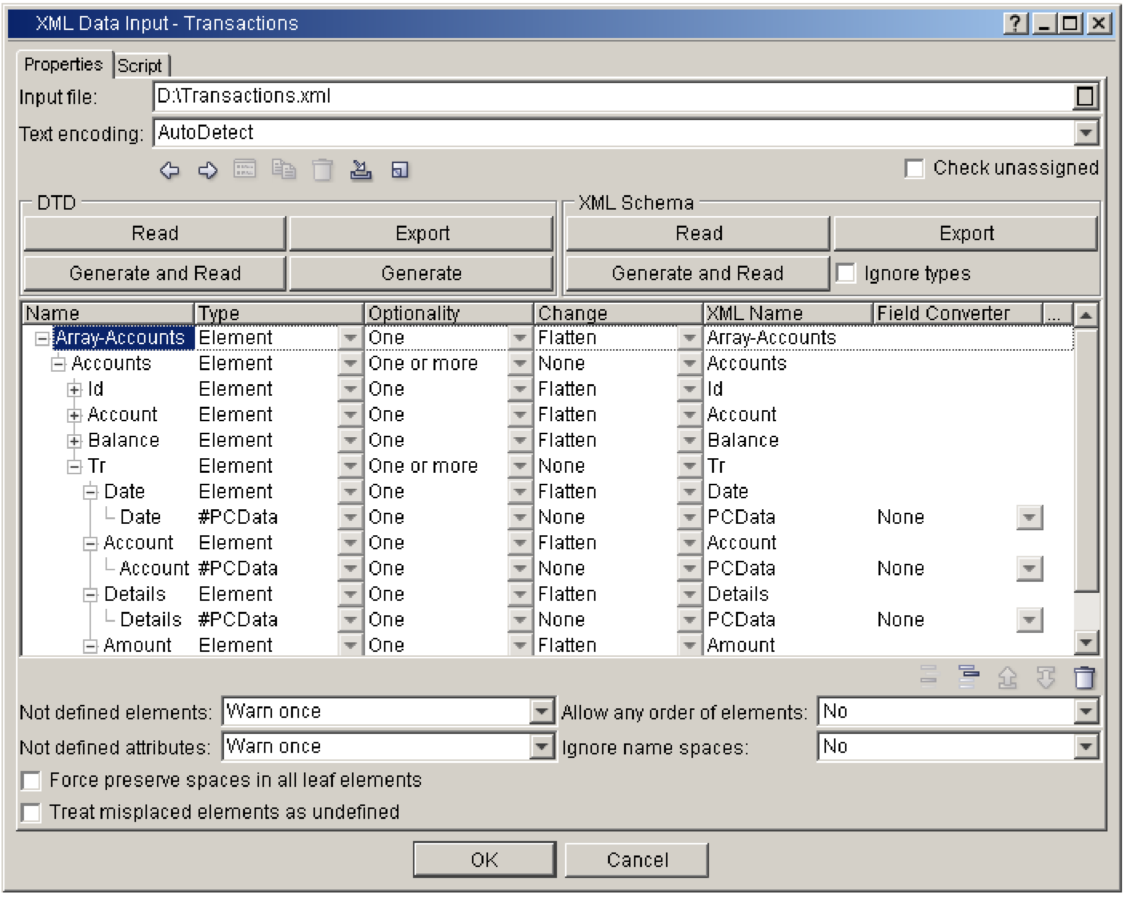
In the **Input file** edit-box, browse □ for the XML file (e.g. Transactions.xml) and click on **Open**. The full path to the file is displayed.

Some knowledge of XML and the content of the XML data file is required to understand the options in this configuration dialog and to be sure that your file is correctly structured in XML. If the data file is correct then it can have a schema generated by the XML Data Input module.

⊕ Generate an XML schema and read it to the module.

In the XML Schema field, click on the **Generate And Read** button.

The different elements of the XML file are displayed:



Input file: D:\Transactions.xml

Text encoding: AutoDetect

□ Check unassigned

□ Ignore types

Name	Type	Optionality	Change	XML Name	Field Converter
Array-Accounts	Element	One	Flatten	Array-Accounts	
Accounts	Element	One or more	None	Accounts	
Id	Element	One	Flatten	Id	
Account	Element	One	Flatten	Account	
Balance	Element	One	Flatten	Balance	
Tr	Element	One or more	None	Tr	
Date	Element	One	Flatten	Date	
Date	#PCData	One	None	PCData	None
Account	Element	One	Flatten	Account	
Account	#PCData	One	None	PCData	None
Details	Element	One	Flatten	Details	
Details	#PCData	One	None	PCData	None
Amount	Element	One	Flatten	Amount	

Not defined elements: Warn once

Allow any order of elements: No

Not defined attributes: Warn once

Ignore name spaces: No

□ Force preserve spaces in all leaf elements

□ Treat misplaced elements as undefined

The preview of the data that has been read from the data file makes it possible to review the file for errors in format.

XML data files are composed from elements that contain the values. These elements can be directly read as variables into the workflow, keeping the same name.

At this point, it is important to check the dependencies of the variables, which are represented in a tree form in the GUI of Inspire Designer. In our example, the variable Accounts is an array of four other variables (Id, Account, Balance and Tr) and the variable Tr (the variable containing transaction values) is an array of another four variables (Date, Account, Details and Amount). Slight changes to the structure can be selected in the **Change** column and some required changes are automatically detected, e.g. *Flatten* is set for elements that only have a one-to-one relationship with the data.

⊕ Compare the structure of the data with the XML data file.

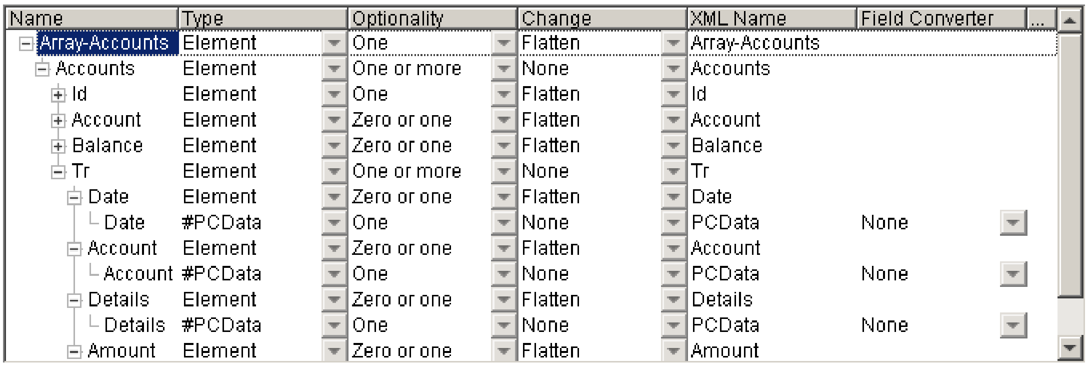
Look at the preview structure and check that the relationship between the elements is in accordance with the requirements of the workflow. Pay special attention to array structures (e.g. *Tr* is an array containing other variables). Adjust the structure if necessary using the options in the **Change** column.

The structure of the variables that will be read from the XML elements is defined.

⊕ Set how values should be expected from the elements of the XML file.

Go through the elements (i.e. those rows with **Type** as *Element*), one by one, and decide if there must always be a value returned by it. If it is always a single-value, select *One* in the **Optionality** column (usually selected by default for elements of one-to-one relationship); if it is always a set of values, then select *One or more* (usually selected by default for arrays); or if it does not have to contain a value, select *Zero or One*.

The **Optionality** of the elements is defined:



Name	Type	Optionality	Change	XML Name	Field Converter	...
⊟ Array-Accounts	Element	One	Flatten	Array-Accounts		
⊟ Accounts	Element	One or more	None	Accounts		
⊞ Id	Element	One	Flatten	Id		
⊞ Account	Element	Zero or one	Flatten	Account		
⊞ Balance	Element	Zero or one	Flatten	Balance		
⊟ Tr	Element	One or more	None	Tr		
⊟ Date	Element	Zero or one	Flatten	Date		
└ Date	#PCData	One	None	PCData	None	
⊟ Account	Element	Zero or one	Flatten	Account		
└ Account	#PCData	One	None	PCData	None	
⊟ Details	Element	Zero or one	Flatten	Details		
└ Details	#PCData	One	None	PCData	None	
⊟ Amount	Element	Zero or one	Flatten	Amount		

The **Optionality** column is for the definition of how values should be expected from each of the elements. For example, some elements might not always be present for a record, therefore, in such cases **Optionality** must be set to *Zero or more* for the reading of elements to succeed.

⊕ Confirm the configuration and return to the Workflow window to add the next module.

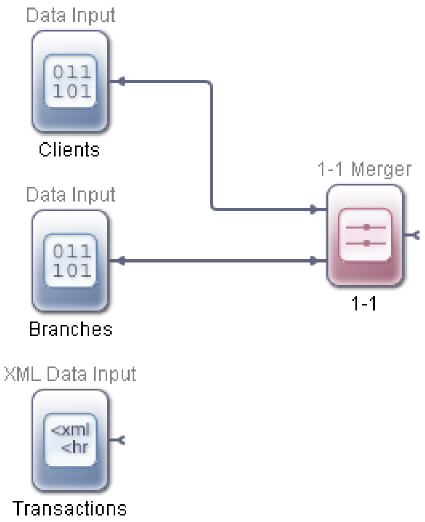
Click on **OK**.

The XML Data Input module closes.

5.2.2.4 1-1 Merge

⊕ Add a module that can read the clients' data from the first data input and match each client to a bank's branch data coming from the second one.

Drag a 1-1 Merge module to the Workflow Area, drop it and connect the top-left connector (input A) to the first data input module (e.g. Clients) and the bottom-left connector (input B) to the second data input module (e.g. Branches).



Configure the 1-1 Merge module using the common data variable that exists in both data inputs.

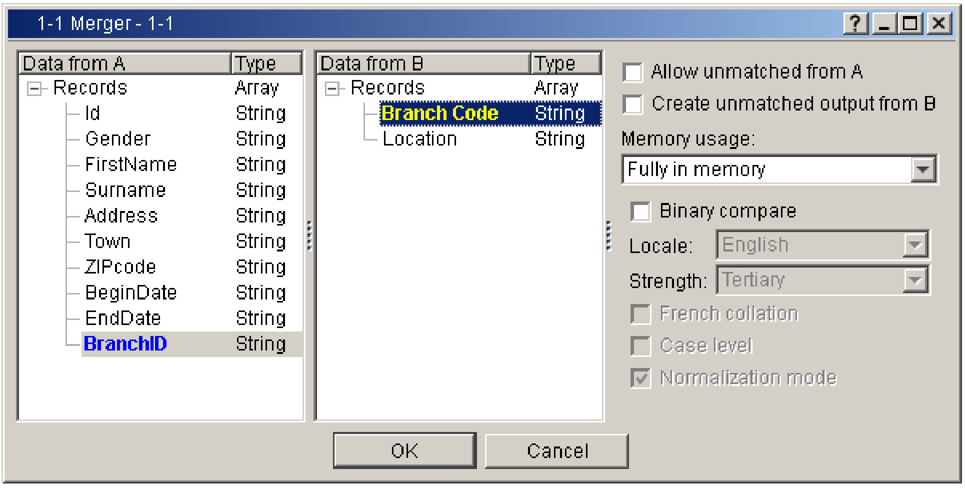
Double-click on the module.

The module opens.

Connect the Branch Code variable read from the Clients data file to the Branch Code variable read from the Branches data file.

In the **Data from A** area, select the variable that contains the common data (e.g. BranchID) from the array and then select the corresponding variable (e.g. Branch Code) in the **Data from B** area.

The two variables are connected and the data from the separate data files will be combined:



1-1 Merger - 1-1

Data from A	Type
Records	Array
Id	String
Gender	String
FirstName	String
Surname	String
Address	String
Town	String
ZIPcode	String
BeginDate	String
EndDate	String
BranchID	String

Data from B	Type
Records	Array
Branch Code	String
Location	String

□ Allow unmatched from A
□ Create unmatched output from B
Memory usage: Fully in memory
□ Binary compare
Locale: English
Strength: Tertiary
□ French collation
□ Case level
☑ Normalization mode

The 1-1 Merge module defines that the data is read from each of the data files and combined to act as one data file. How this is done is affected by the option **Memory usage** which determines what information will be stored within Inspire Designer when it is processing the data. The fastest processing can happen if one of the data files is wholly loaded to the application memory, but often this is not desirable due to live data or the larger file size it incurs. In our example, we only have a very small TXT file as the second data input, therefore it is more efficient if *Fully in memory* is selected from the combo-box here. This determines that all of the **Data from B** will be loaded to the application memory when the workflow is processed.

⊕ Confirm the configuration and return to the Workflow window to add the next module.

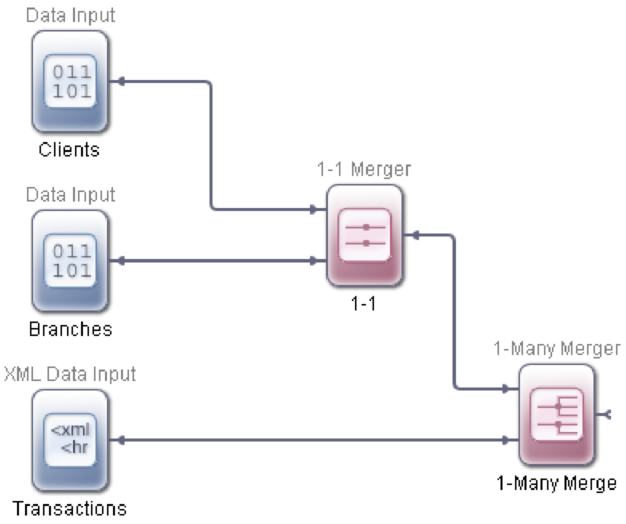
Click on **OK**.

The 1-1 Merge module closes.

5.2.2.5 1-Many Merge

⊕ Add a module that can read the data that was combined by the 1-1 Merge module and match each client to their transactions that are listed in the XML data.

Drag a 1-Many Merge module to the Workflow Area, drop it and connect the top-left connector (input A) to the 1-1 Merge module and the bottom-left connector (input B) to the XML Data Input module (e.g. Transactions).



⊕ Configure the 1-Many Merge module using the variables that have been read from a data field that is common to both data inputs:

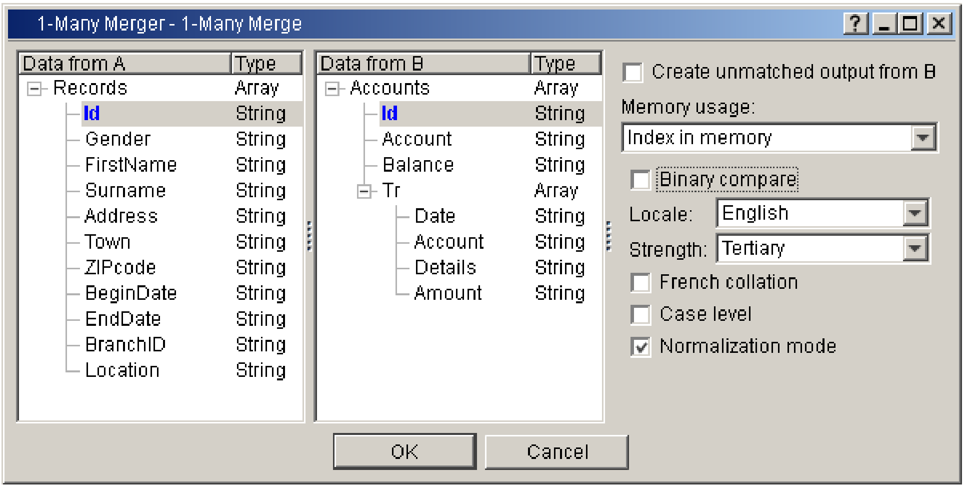
Double-click on the module.

The module opens.

⊕ Connect the account number variable, that is read from the account number field of the data file that contains the clients information, to the account number variable that is read from the XML file.

In the **Data from A** area, select the variable that contains the common data (e.g. Id) from the array and then select the corresponding variable (e.g. Id) in the **Data from B** area.

The two variables are connected and the data from the separate data files will be combined:



1-Many Merger - 1-Many Merge

Data from A	Type
Records	Array
Id	String
Gender	String
FirstName	String
Surname	String
Address	String
Town	String
ZIPcode	String
BeginDate	String
EndDate	String
BranchID	String
Location	String

Data from B	Type
Accounts	Array
Id	String
Account	String
Balance	String
Tr	Array
Date	String
Account	String
Details	String
Amount	String

□ Create unmatched output from B

Memory usage: Index in memory

□ Binary compare

Locale: English

Strength: Tertiary

□ French collation

□ Case level

☑ Normalization mode

Like the 1-1 Merge module, the 1-Many Merge module defines that the data is read from each of the data inputs and combined to act as one data file. It differs only in the fact that an array will be created in the first input (input A) and that array will contain all of the variables from the other input (input B). All of the data from input B then becomes dependent on the one linked variable (e.g. the account identification variable *Id*).

Processing is again affected by the option **Memory usage**. In this example, we have a larger amount of information in the XML input file (input B), so it is better to select *Index in memory* from the combo-box. This determines that only an index of the structure of the selected **Data from B** will be loaded to the application and the data files will only be read when the workflow is run.

⊕ Confirm the configuration and return to the Workflow window to add the next module.

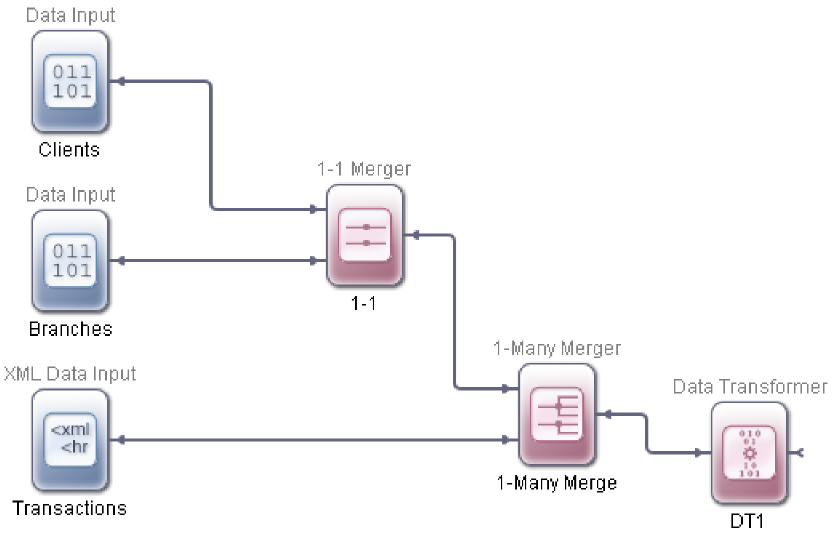
Click on **OK**.

The 1-Many Merge module closes.

5.2.2.6 Data Transformer 1

⊕ Add a module that can convert some of the variables of string data-type into values of other data-types that reflect their nature (e.g. recognize date strings as time values or amount strings as money values).

Drag a Data Transformer module to the Workflow Area, drop it and connect to the 1-Many Merge module.



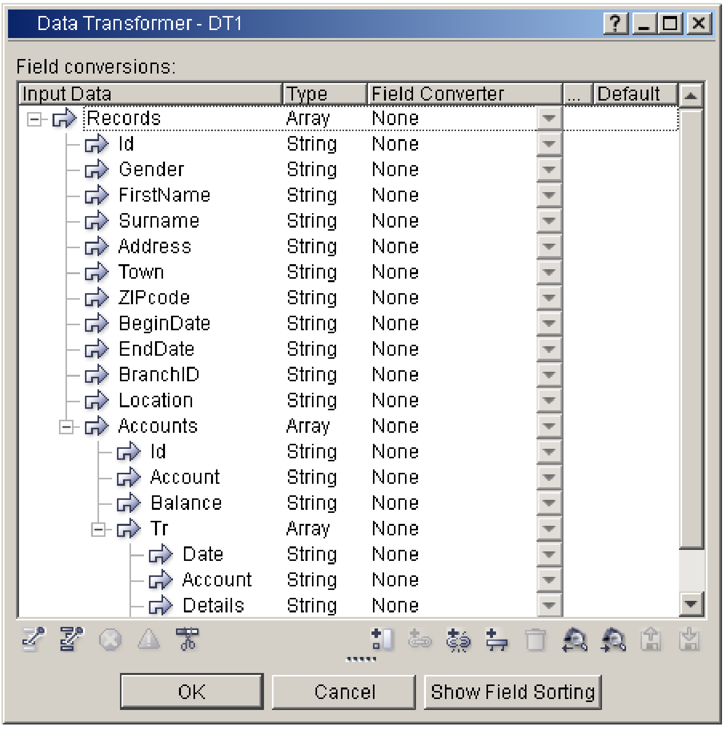
Data Transformers are often required to convert data into another data-type because values need to be in their true form in order to manipulate them in the correct way, e.g. numbers must be of a numerical data-type if they are to be used in a calculation.

The Data Transformer can also add 'linked' data variables that manipulate the input data and return values to a new variable that can be used further in the workflow.

⊕ Configure the Data Transformer.

Double-click on the module.

The module opens.



Field conversions:

Input Data	Type	Field Converter	...	Default
Records	Array	None		
Id	String	None		
Gender	String	None		
FirstName	String	None		
Surname	String	None		
Address	String	None		
Town	String	None		
ZIPcode	String	None		
BeginDate	String	None		
EndDate	String	None		
BranchID	String	None		
Location	String	None		
Accounts	Array	None		
Id	String	None		
Account	String	None		
Balance	String	None		
Tr	Array	None		
Date	String	None		
Account	String	None		
Details	String	None		

All of the variables from the combined data inputs are listed according to their dependency, along with their current *data-type*. In our example, all of the value-returning variables are in *String* (basic text) data-type. Looking at the data and comparing it to how it should appear on the account statement helps us to decide which variables need to be transformed. We can also decide if there are any other manipulations wanted that should be performed using the input data. Some of the variables contain string values that represent values of another data-type and so conversions are needed and also transaction related data is required to be recognizable as positive or negative.

In this Data Transformer, we will define the following conversions and manipulations:

1. String to Date
2. String to Numerical Value
3. Transaction amount to create linked variables 'Debit' and 'Credit''
4. Transaction amount to create a linked Boolean variable 'IsCredit'

Parse DateTime (1)

In order to conform to recognizable date and time standards, strings that represent dates can be converted to date and time *data-type*.

⊕ Convert each of the variables with date values into variables of date data-type.

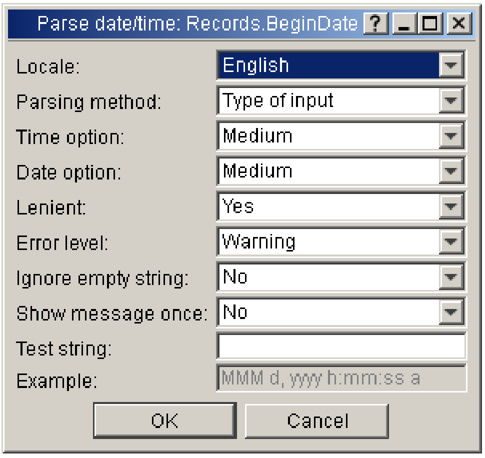
For each variable that is intended to be a date (e.g. BeginDate, EndDate and Date [date of transaction]) select *Parse DateTime-DateTime* from the drop-down menu in the column **Field Converter**.

The conversion is set.

⊕ Configure how the date should be read.

Click on the **3 Dots** … icon that has appeared in the **...** column.

The ParseDateTime dialog opens:



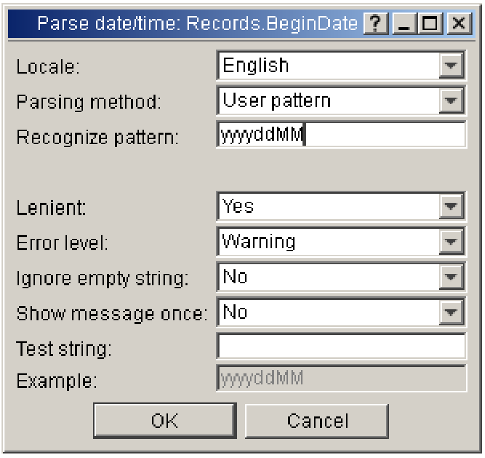
Parse date/time: Records.BeginDate

Locale: English
Parsing method: Type of input
Time option: Medium
Date option: Medium
Lenient: Yes
Error level: Warning
Ignore empty string: No
Show message once: No
Test string:
Example: MMM d, yyyy h:mm:ss a

It is important to look carefully at the format of the date in the data file before defining how it should be recognized, i.e. identify which part of the value is the day, which part is the month and which part is the year.

⊕ Define the method by which the variable content will be recognized as a date. Each character from the incoming data should be substituted with a format character identifying which part of the date it is.

In the **Parsing Method** combo-box select *User pattern* from the drop-down menu. In the **Recognize Pattern** edit-box that appears enter which character from the input data is what part of the date. Use the substitution characters that are shown in the help note that appears (e.g. y = Year, M = Month, d = Day can give a pattern yyyyddMM).



Parse date/time: Records.BeginDate

Locale: English
Parsing method: User pattern
Recognize pattern: yyyyddMM
Lenient: Yes
Error level: Warning
Ignore empty string: No
Show message once: No
Test string:
Example: yyyyddMM

Conversions can be copied (<**Ctrl**>+<**C**>) and pasted (<**Ctrl**>+<**V**>), once they are configured, to other variables in the Data Transformer module. This speeds up assigning multiple transformations of the same data-type to different variables.

⊕ Confirm the configuration and return to the Data Transformer.

Click on **OK**.

The ParseDateTime dialog closes.

BC.-Double (2)

In order to perform calculations with numerical entries from a data file, they need to be converted to numerical data-type such as double.

⊕ Convert variables that are monetary values so that they will be recognized as numerical.

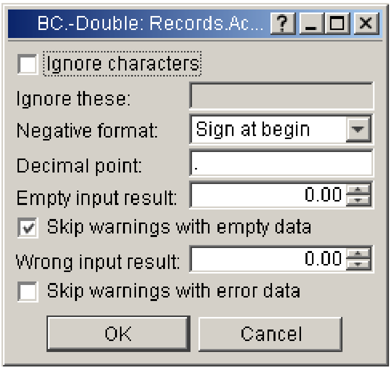
Select each variable that represents a money value (e.g. Balance [account balance] and Amount [value of transaction]) and in the column **Field Converter** select *BC.-Double.* from the drop-down menu.

The conversion is set.

⊕ Configure how monetary amounts should be read.

Click on the **3 Dots** ... icon that has appeared in the **...** column.

The BC.-Double dialog opens:



BC.-Double: Records.Ac...

□ Ignore characters
Ignore these:
Negative format: Sign at begin
Decimal point: .
Empty input result: 0.00
☑ Skip warnings with empty data
Wrong input result: 0.00
□ Skip warnings with error data

The values of the attributes shown in the screenshot are the default settings. They happen to suit the data used in our example file so we will not adjust them. However, pay attention to them as you may require your **Decimal point** to be a comma "," or the **Negative format** to be displayed in brackets (e.g. as (280.00), select *Round bracket* for this appearance) or a different amount to be displayed in the case of no value being given (can be set in the **Empty input result** combo-box).

⊕ Confirm the configuration and return to the Data Transformer.

Click on **OK**.

The BC.-Double dialog closes.

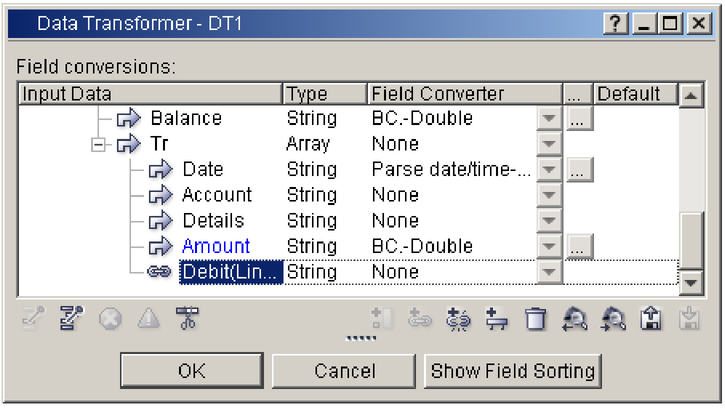
Linked Variables 'Debit' and 'Credit' (3 & 4)

For the purpose of making calculations with negative and positive transaction values, we require 'credits' and 'debits' to be written as additional, separate variables. They can then be easily referenced.

⊕Create a new linked variable that is based upon the transaction values variable.

Select the transaction value variable (e.g. Amount) and click on the **Link** icon.

A new data variable is created:



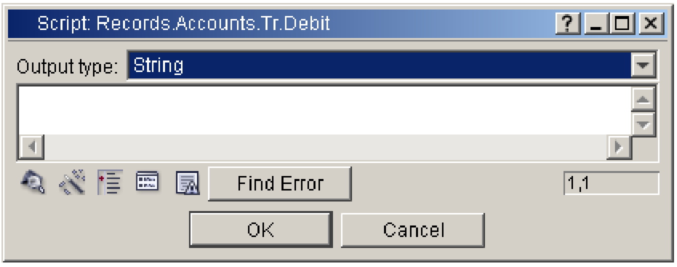
Field conversions:

Input Data	Type	Field Converter	...	Default
Balance	String	BC.-Double	...	
Tr	Array	None		
Date	String	Parse date/time-...	...	
Account	String	None		
Details	String	None		
Amount	String	BC.-Double	...	
Debit(Lin...	String	None		

⊕ Assign a script to define the content of the variable.

In the **Field Converter** column of the new variable, select *Script* from the drop-down menu and click on the **3 Dots** icon that appears in the **...** column.

The Script dialog opens:



Output type: String

As record variables can be accessed in the script using the name that was given to the corresponding field in the data input, it is quite simple to add short scripts. In our example, we want to manipulate the data in the transaction value variable (e.g. Amount) which first needs to be converted to double data-type (toDouble). Therefore, we use the corresponding variable name plus the conversion that should be made (e.g. `Amount.toDouble`) in the script.

⊕ Write a script, based on the commands `if` and `return`. The script checks if a transaction value is negative and, if so, returns it to the new variable.

Copy the following script to the script area:

```
if (Amount.toDouble()<0) return Amount.toDouble();
```

The new variable content is defined by a script that states "if the variable (value) is less than 0, return the variable (value)".

Remember to use the actual name of the variable that is being manipulated as well as the necessary conversion of this variable, using the formula `VariableName.toConversion`. E.g., if you have the variable `Amount` and want to use the conversion *Double*, you would have to write `Amount.toDouble` into the script.

Remember: You can validate your script using the **Find Error** button.

⊕ Define that the format of the data in this variable also should be numerical.

In the **Output type** combo-box select *Double*.

The data in the variable will be of double data-type.

⊕ Confirm the configuration and return to the Data Transformer.

Click on **OK**.

The Script dialog closes.

⊕ Create a new linked variable, based upon the transaction values variable. Write a script that checks if the transaction values are positive and that, if this is the case, will return them to the new variable.

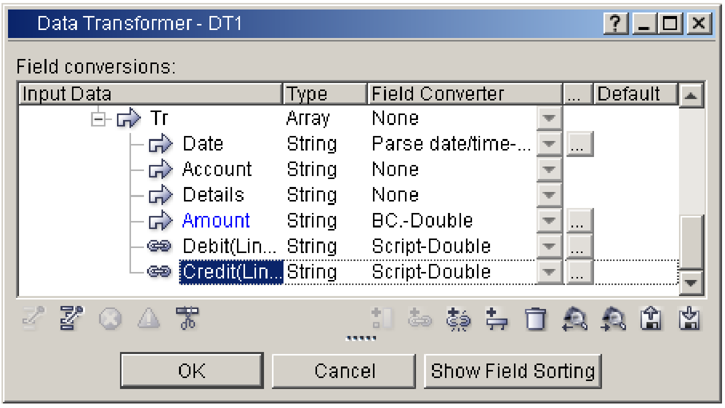
Repeat the previous 5 steps, writing the same script with > (greater than) instead of < (less than).

Another new variable is defined by a script that states "if the variable (value) is greater than 0, return the variable (value) and give it double data-type ".

⊕ Rename the two new variables so that they will be easily recognizable further in the workflow.

Select each variable in the Data Transformer configuration dialog and press <**F2**> to be able to type in replacement names (e.g. *Debit* and *Credit*).

The creation of two new linked variables, that contain either Debits only or Credits only, is complete.



Data Transformer - DT1

Field conversions:

Input Data	Type	Field Converter	...	Default
Tr	Array	None		
Date	String	Parse date/time-...	...	
Account	String	None		
Details	String	None		
Amount	String	BC.-Double	...	
Debit(Lin...	String	Script-Double	...	
Credit(Lin...	String	Script-Double	...	

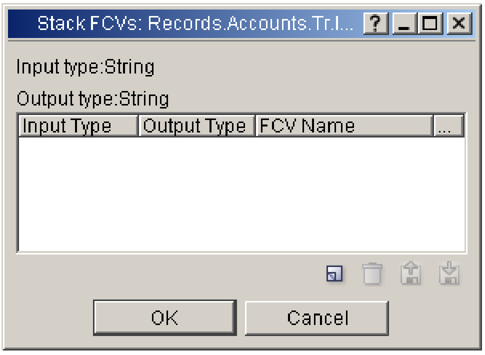
Linked Variable 'IsCredit' (5)

For the specific purpose of indicating whether a transaction value (e.g. the *Amount* variable) is positive or negative we require a new variable with only true or false entries (i.e. the Boolean values 1 or 0) in answer to the question 'is the value positive?'. This cannot be done directly from a string input, as it is not yet possible to recognize that it is even a number. Therefore, we must assign a series of conversions.

⊕ Create a new linked variable that is also based upon the transaction values variable and assign a series of variable conversions (a Stacked FCV) to define the format of its content.

Select the transaction value variable (e.g. Amount) and click on the **Link** icon. Then, in the **Field Converter** column of the new variable, select *Stack FCVs* from the drop-down menu. Click on the **3 Dots** icon that appears in the **...** column.

The Stack FCVs dialog opens.



Stack FCVs: Records.Accounts.Tr.l...

Input type:String

Output type:String

Input Type	Output Type	FCV Name	...

A stacked FCV puts two or more conversions in series, allowing for format transformations that cannot be done in only one step.

⊕ Assign the first conversion of the stack, a conversion to double numerical data-type.

Click on the **Create New** icon and in the **Select Type of New Converter** dialog that appears select *BC.-Double*.

A new conversion appears as a row in the FCV Area.

Remember [85]: The BC.-Double dialog can be opened to configure the conversion by clicking on the **3 Dots** icon that has appeared in the **...** column.

⊕ Assign the second conversion of the stack, a scripted conversion to a Boolean value.

Click on the **Create New** icon and in the **Select Type of New Converter** dialog that appears select *Script*.

Another new conversion appears as a row in the FCV Area.

⊕ Open the script dialog and enter a short script that declares that if the transaction value is positive then the result of the new linked Boolean variable 'IsCredit' is true.

Click on the **3 Dots** icon that appears in the **...** column to open the Script dialog.

Select **Output type** as *Bool* and enter the following script in the script area:

```
Input>0
```

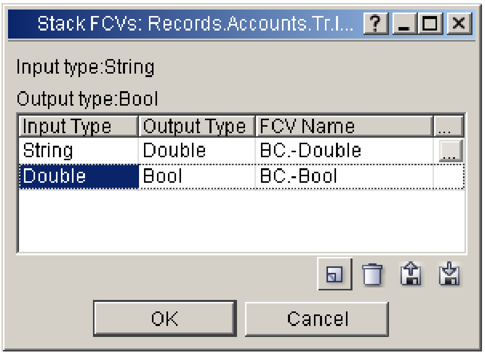
The new variable content is defined by a script that declares that the Boolean is true (returns the value *1*) if the value received from the previous conversion to a numerical format is greater than 0.

`Input` can be used instead of the name of the variable (e.g. Amount) because, in this context, it is the value from the previous conversion that is being manipulated.

⊕ Confirm the script.

Click on **OK**.

The Script dialog closes to reveal the configured Stack of FCVs:



Input type:String

Output type:Bool

Input Type	Output Type	FCV Name	...
String	Double	BC.-Double	...
Double	Bool	BC.-Bool	

⊕ Confirm the configuration and return to the Data Transformer.

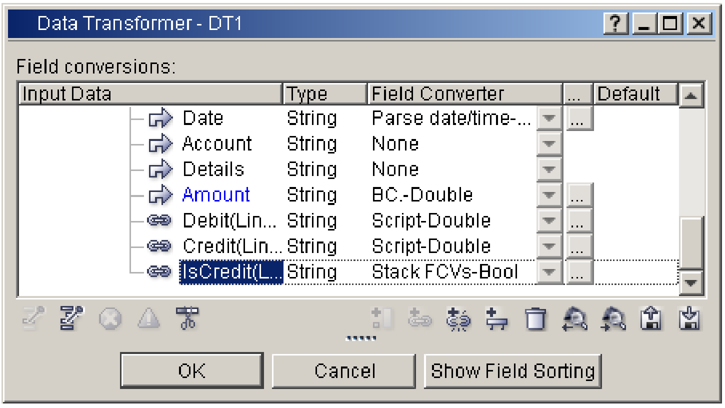
Click on **OK**.

The Stack FCVs dialog closes.

⊕ Rename the new variable so that it will be easily recognizable further along the workflow.

Select the variable in the **Input Data** column and press <**F2**> to be able to type in a replacement name (e.g. *IsCredit*).

The creation of a new stacked-conversion linked variable is complete.



Field conversions:

Input Data	Type	Field Converter	...	Default
Date	String	Parse date/time-...	...	
Account	String	None		
Details	String	None		
Amount	String	BC.-Double	...	
Debit(Lin...	String	Script-Double	...	
Credit(Lin...	String	Script-Double	...	
IsCredit(L...	String	Stack FCVs-Bool	...	

⊕ Confirm the configuration of the Data Transformer and return to the Workflow window to add the next module.

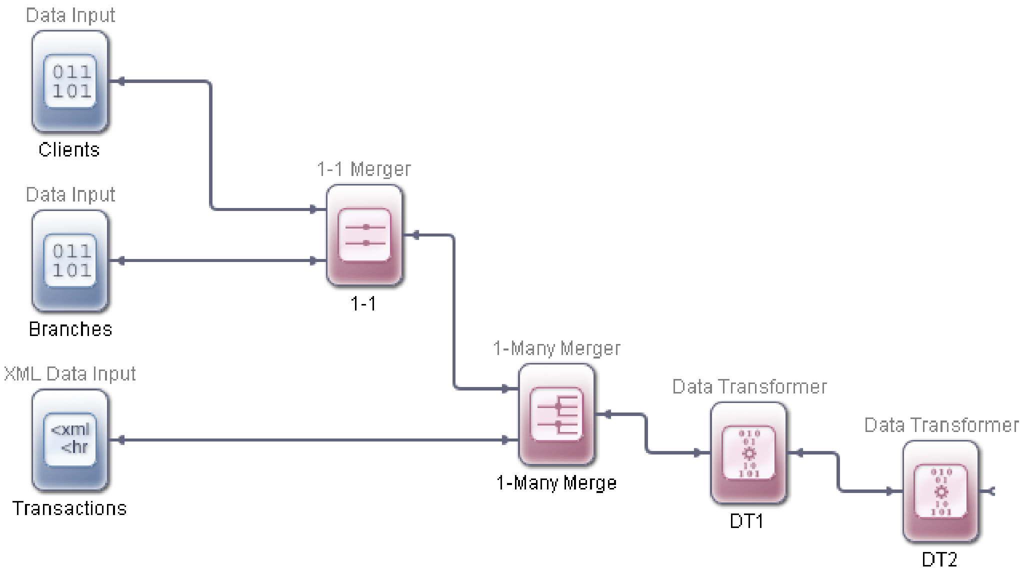
Click on **OK**.

The Data Transformer closes.

5.2.2.7 Data Transformer 2

⊕ Add a module to calculate totals from the values of the new 'linked' variables.

Drag another Data Transformer module to the Workflow Area, drop it and connect it to the previous one.



The Data Transformer module can also make simple calculations with the data and write the results into another new variable. In this example, we would like to be able to sum up all of the debits and, separately, all of the credits, to add the sums as extra information in our bank statement. This can be done easily by selecting the according variable and clicking on the **New Special Field** icon. However, to be able to calculate a sum with this icon, the variable which is selected needs to be of a numerical *data-type* (e.g. double).Therefore, the conversions that we set for the transaction values in the previous Data Transformer module need to have been executed. This is why a second Data Transformer module is required.

In this section, we will create the following calculated variables :

1. TotalDebit – The sum of all the negative transactions on an account
2. TotalCredit – The sum of all the positive transactions on an account
3. ClosingBalance – The sum of the opening balance, the negative transactions and the positive transactions on an account

⊕ Configure the Data Transformer.

Double-click on the module.

The module opens with all of the variables (including the new linked variables) and their dependencies listed with their new *data-type* in the **Type** column.

TotalDebit and TotalCredit (1 & 2)

⊕ Calculate the total for each record of the newly created 'debit transactions only' variable and have the result written as a new variable.

Select the variable (e.g. Debit) and click on the **New Special Field** icon that can be found at the bottom of the Data Transformer dialog. Then, in the Select Data Operation dialog that appears, select *Sum* as the **Type**.

The new calculated variable is created as a new row in the Data Transformer.

⊕ Calculate the total for each record of the newly created 'credit transactions only' variable and have the result written as a new variable.

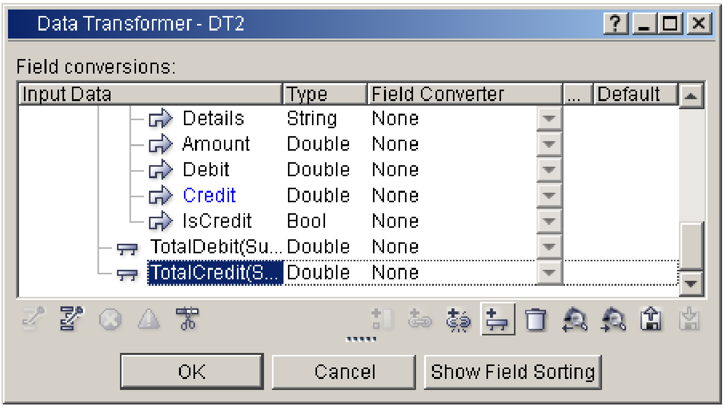
Repeat the previous step, this time selecting the 'credit transactions only' variable (e.g. Credit).

The variable that returns the sum of credit transactions is created.

⊕ Rename the new variables so that they will be easily recognizable further in the workflow.

For each of the variables, select the **Input Data** column and press <**F2**> to be able to type in replacement names (e.g. *TotalDebit* and *TotalCredit*).

The creation of the new calculated variables is complete.



Field conversions:

Input Data	Type	Field Converter	...	Default
Details	String	None		
Amount	Double	None		
Debit	Double	None		
Credit	Double	None		
IsCredit	Bool	None		
TotalDebit(Su...	Double	None		
TotalCredit(S...	Double	None		

ClosingBalance (3)

In our example, it will be useful to have another variable that calculates the closing balance of the account, i.e. a variable that uses the existing opening balance variable and the two new total variables to calculate a closing balance that can be used at the bottom of the statement.

⊕ Create a new linked variable that will use a short script to return the end balance of the account by summing the new 'Sum' variables plus the opening balance of the account.

Select the account's opening balance variable (e.g. Balance) and click on the **Link** icon. Then, in the **Field Converter** column of the new variable, select *Script* from the drop-down menu. Click on the **3 Dots** icon that appears in the **...** column.

The script dialog opens.

⊕ Write a simple script that will return the sum of the two newly created variables and balance.

Copy the following script to the script area:

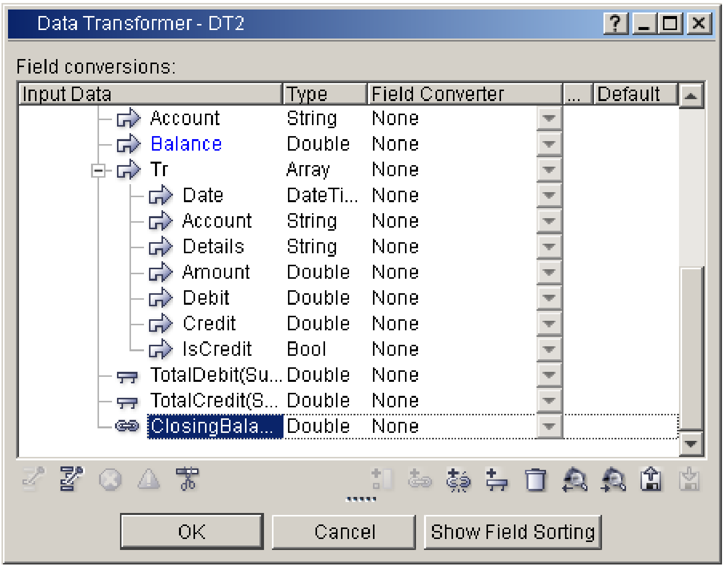
```
TotalDebit+TotalCredit+Balance
```

Remember to use the actual name of the variables that are being manipulated. Change the **Output type** to *Double*.

The new linked variable is defined by a script that states "calculate the sum of the opening balance of account + the total of debits + the total of credits".

Select the new variable in the **Input data** column and press <**F2**> to be able to type in replacement names (e.g. Closing Balance).

Now there are three variables that calculate transactions that will be useful in the account statement.



Data Transformer - DT2

Field conversions:

Input Data	Type	Field Converter	...	Default
Account	String	None		
Balance	Double	None		
Tr	Array	None		
Date	DateTi...	None		
Account	String	None		
Details	String	None		
Amount	Double	None		
Debit	Double	None		
Credit	Double	None		
IsCredit	Bool	None		
TotalDebit(Su...	Double	None		
TotalCredit(S...	Double	None		
ClosingBala...	Double	None		

⊕ Confirm the configuration of the Data Transformer and return to the Workflow window to add the next module.

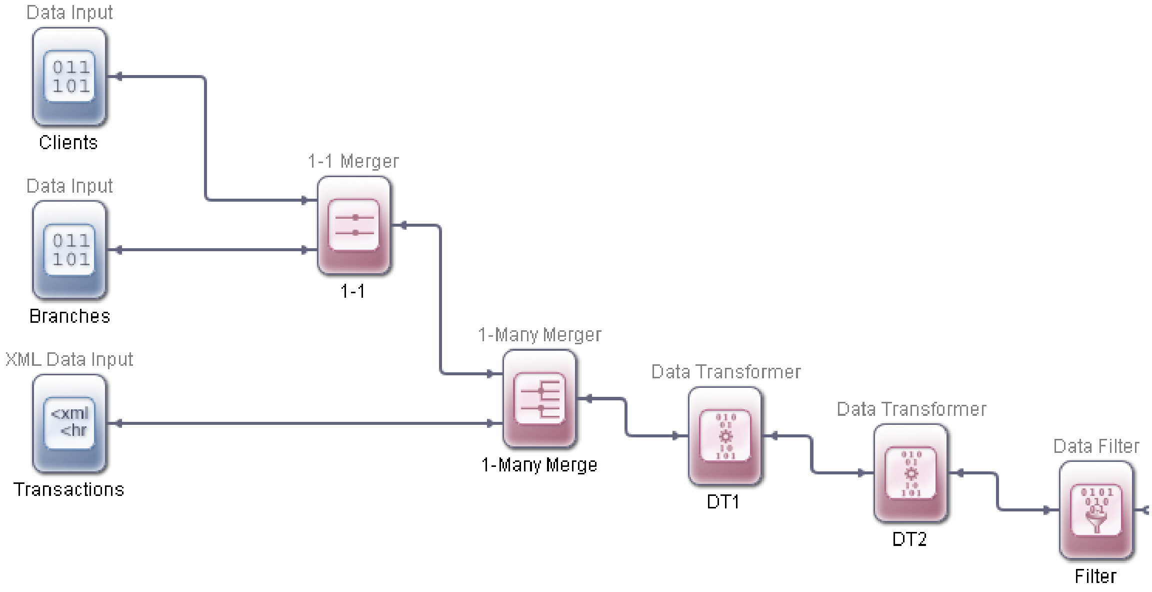
Click on **OK**.

The Data Transformer closes.

5.2.2.8 Data Filter

⊕ Add a module to block the records of clients that have not made any transactions, so that a bank statement will not be printed for them.

Drag a Data Filter module to the Workflow Area, drop it and connect it to the second Data Transformer.



⊕ Configure the Data Filter.

Double-click on the module.

The module opens.

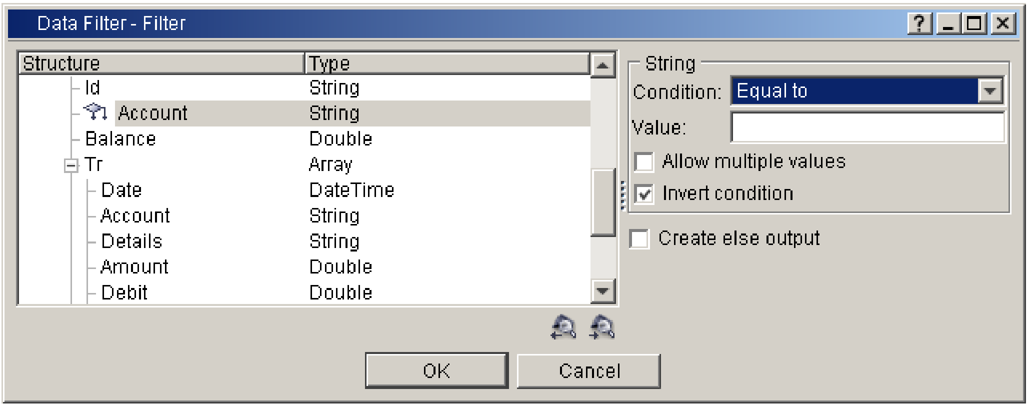
The current data variables are listed with their *data-types*.

The Data Filter can allow or disallow records to continue through the workflow depending on the values of selected variables. A common application for this is to remove variables that are empty. In our example, we have a client in our clients data file (Clients.csv) that does not have any matching transactions in the transactions data file (Transactions.xml). Now that all of the data from the separate inputs has been combined, all of the variables that have been created exist for all clients. But if a client had no data in the transactions file, then their transaction related variables will be empty at this stage. Therefore, by selecting one of the transaction variables and making a condition that declares "if this variable is empty then do not let the record containing it pass" we can prevent this client's data from reaching the document design stages of the workflow.

⊕ Block any records that have empty transaction variables so that they are prevented from being processed further.

Select a transaction related variable that is unique to the transaction data input and in String format, such as the account number (e.g. Account), and from the **Condition** combo-box menu select *Equal to*. Then, leave the **Value** edit-box empty and check the **Invert condition** check-box.

The filtered variable is defined:



Data Filter - Filter

Structure	Type
Id	String
Account	String
Balance	Double
Tr	Array
Date	DateTime
Account	String
Details	String
Amount	Double
Debit	Double

String
Condition: Equal to
Value:
☐ Allow multiple values
☑ Invert condition
☐ Create else output

For our selected string variable, by leaving the edit-box **Value** empty we are actually determining that the variable has to be *equal to* an empty variable in order for the condition we are creating to be true. Furthermore, we want to use the **Invert condition** functionality which has the following effect: if the condition is met (returns *TRUE*) then the outcome is actually false. A negative outcome blocks the record from passing further.

⊕ Confirm the configuration of the Data Filter and return to the Workflow window to add the next module.

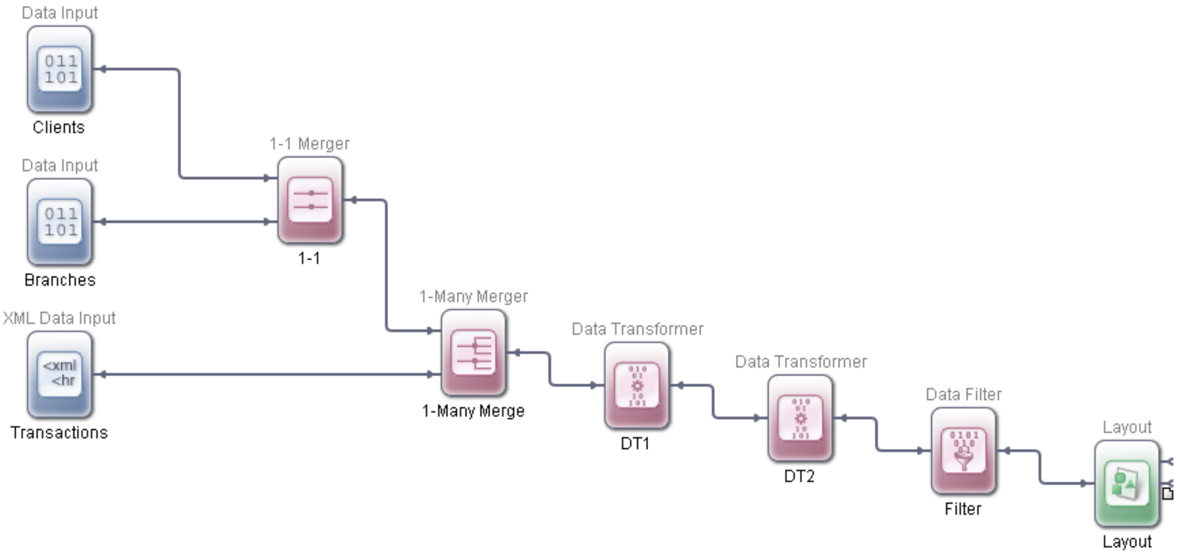
Click on **OK**.

The Data Filter closes.

5.2.2.9 Layout

⊕ Add a Layout module in which we will design the pages of the account statement.

Select Layout from the Module Tree, drag it to the Workflow Area, drop it and connect it to the Data Filter module.

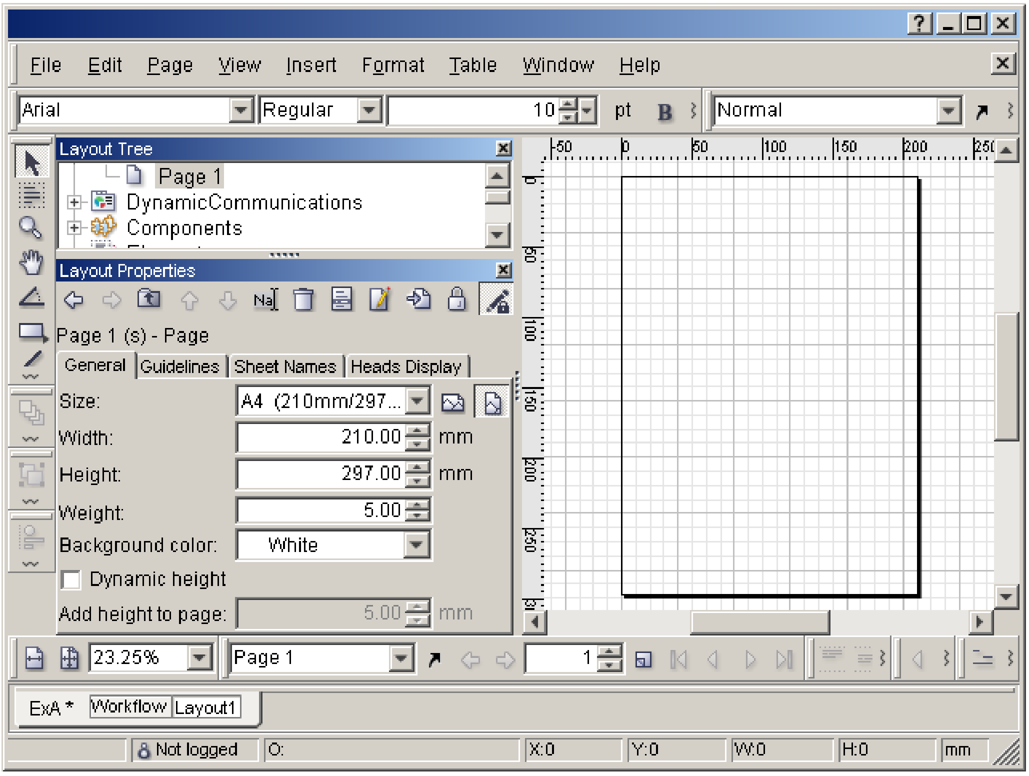


⊕ Design a bank statement that will be personalized and sent to each of the clients. The content of this statement is made up of the data that is being fed from the workflow modules that has been already defined. The statement should include a list of transactions with an unknown length.

Double-click on the Layout module.

The Layout window opens.

Layout Window



Page 1 (s) - Page

Size: A4 (210mm/297...

Width: 210.00 mm

Height: 297.00 mm

Weight: 5.00

Background color: White

□ Dynamic height

Add height to page: 5.00 mm

Remember: We need to create the following features for the Account Statement:

1. Two Pages
2. Logo and Template Shapes
3. Barcode
4. Client Address Area
5. Statement Details Area
6. Account Details Area
7. Account Statement Body
8. Correctly Formatted Data

Two Pages (1)

In this example, we have two page designs to create: One for the first page of the statement and another for the continuation pages, for when there are more transactions than can fit on the first page. Certain features of these pages are the same and therefore it proves a little easier to work on the separate designs in parallel.

⊕ Create two separate pages so that two possible page designs can be chosen from.

Right-click on the Pages family and select **Insert Page** from the context menu.

The new page is inserted into Layout and can be seen as an object in the Layout Tree (called Page 2 by default).

As the Layout module opens with a blank page already created, we only need to create one more for our example. The new page is selected (opened) when it is created.

From now on in the example, you can navigate between both of the pages using the Layout Tree.

Remember: Rename the pages (e.g. to FirstPage and NextPage). The pages will be referred to under these names in the further course of this example.

For the pages of the Layout Design we must further define the following attributes:

- Page Order
- Sheet Names

Page Order

The page order for our example should be defined in the following way: Always print the FirstPage for each new account and if the account's statement details do not fit on to FirstPage, then NextPage must be selected and then, if further required, repeated.

Remember: Page order definition starts with the properties of the Pages family.

⊕ Define that the repetition of pages is determined by each new account.

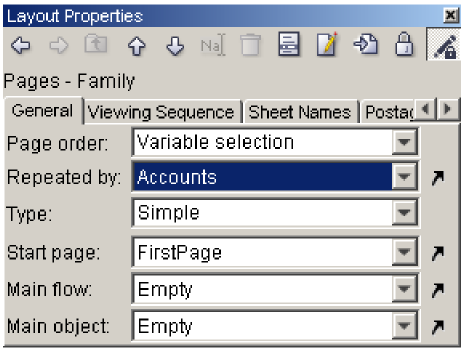
Select the Pages family and, in the **Page order** combo-box on the General tab of its properties panel, choose *Variable selection*. Then in the **Repeated by** combo-box select the variable that contains the account details (e.g. *Accounts*), i.e. the array that is being read from the XML file.

Pages will be repeated for each occurrence of the array that contains new account details.

⊕ Define that the page order starts with FirstPage.

In the **Start page** combo-box select the page that should be the first of the repetition series (e.g. *FirstPage*).

The first page of the page repetition is defined:



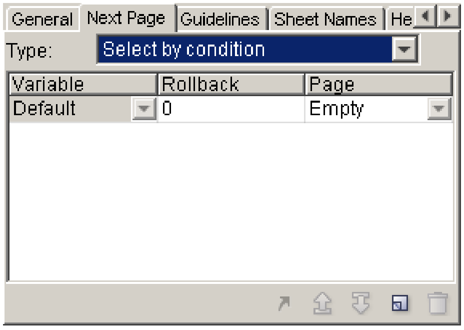
Page order: Variable selection
Repeated by: Accounts
Type: Simple
Start page: FirstPage
Main flow: Empty
Main object: Empty

Remember: Variables are calculated by Inspire Designer when a workflow is processed. The System Variables are always present in Layout, and can be used both to return values to the design or to select objects such as pages. One useful system variable is Overflow, a Boolean variable that returns true if the content of a page overflows. This can then be used as a detector that determines when to select a page that is designed to receive overflowing content. In our example, NextPage is just that: designed to receive the contents of FirstPage (or even of another occurrence of NextPage) if it overflows. Therefore, we can use the Overflow variable to select the next page for both FirstPage and NextPage.

⊕ For both of the pages, define that the page order will be determined by a Boolean variable.

Return to each of the two pages in the Layout Tree and, on the Next Page tab of its properties panel, select *Select by condition* from the **Type** combo-box.

A field for setting conditions appears on the tab:



Type: Select by condition

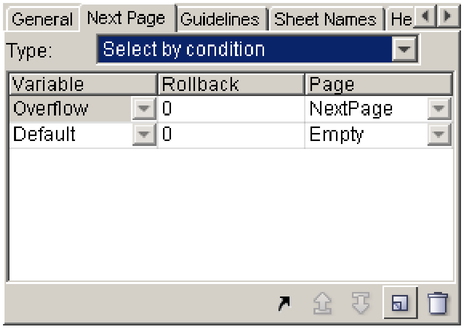
Variable	Rollback	Page
Default	0	Empty

The default next-page setting of *Empty* determines that no page will follow and, in practice, the job will continue with the first page of the design and start to print the details from the next record. Therefore, if the content of the page did overflow, that content which did not fit on the page with the default next-page setting of *Empty* would not be printed.

⊕ Select that the Boolean variable Overflow will be used and that, if it returns true, then the NextPage will be used to display the overflowing data.

Click on the **Create New** icon and, in the new row that appears on the tab as the first row, select the *Overflow* variable from the drop-down menu of the **Variable** column. Then select *NextPage* from the drop-down menu of the **Page** column.

The next page for both pages is determined as NextPage, to be selected only if the page overflows:



Type: Select by condition

Variable	Rollback	Page
Overflow	0	NextPage
Default	0	Empty

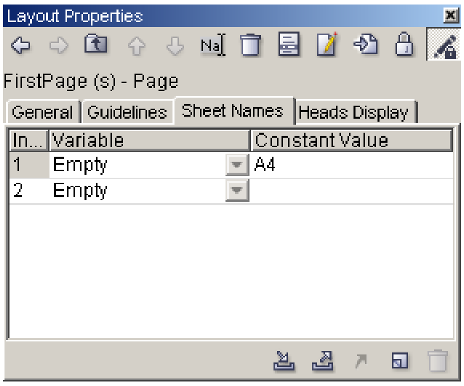
Sheet Names

This design has two separate page designs, but they are both intended to be printed on the same medium (e.g. A4 size paper). Because of this, the first sheet name should be the same for both pages.

⊕ Assign a single sheet name to both of the individual pages so that they can be recognized as pages that require the same medium when being printed.

For each page, select it and click on the Sheet Names tab and in the **Constant Value** field of the first row enter an appropriate name (e.g. *A4*) for sheet name 1.

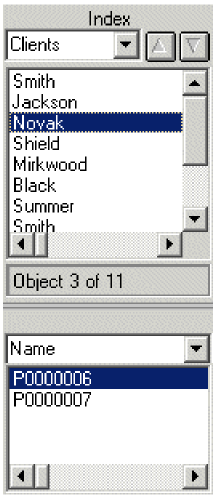
A sheet name is assigned to the pages of the design:



In...	Variable	Constant Value
1	Empty	A4
2	Empty	

Next, we will set up two sheet names that will apply to the actual sheets that will be printed, so that we can differentiate between them and group related sheets together in groups. In our example, we would like to group sheets for each client and, if that client has more accounts, group these under the client's name.

Using a sheet name, we can differentiate between records by returning the surname of the client that a sheet or a set of sheets has been printed for. And, because one client can have more than one account, we could also use another sheet name to identify each of the accounts that the sheets have been printed for. Later in this example we will use these sheet names to create an index in the AFP spool file that we will produce.



Index

Clients

- Smith
- Jackson
- Novak
- Shield
- Mirkwood
- Black
- Summer
- Smith

Object 3 of 11

Name

- P0000006
- P0000007

Figure 5.13 An Example of an AFP Spool File Index Created Using Sheet Names (Viewed in an AFP Browser)

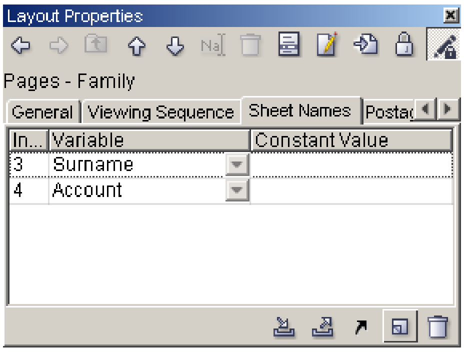
Remember: Sheet names that apply to all pages of the design should be applied via the Pages family in the Layout Tree.

⊕ Create two different sheet names and link them to variables that are read from the records of data.

Select the Pages family and, on the Sheet Names tab of its properties panel, click on the **Create New** icon twice to add two rows to the sheet names area. Two rows for sheet name definition are added to the tab giving the new sheet names an index of 3 and 4.

In the **Variable** column of those rows, select required variables that relate to each client's record (e.g. sheet name *3* equals *Surname*) and to each client's accounts (e.g. sheet name *4* equals *Account* from the Accounts array).

Each sheet that will be printed will have an index value that is related to the client that it belongs to and the account that it belongs to:



Layout Properties

Pages - Family

In...	Variable	Constant Value
3	Surname	
4	Account	

Remember: Sheet names with index numbers 1 and 2 are applicable only for individual pages. The next sheet name indexes are given in order of creation.

Logo and Template Shapes (2)

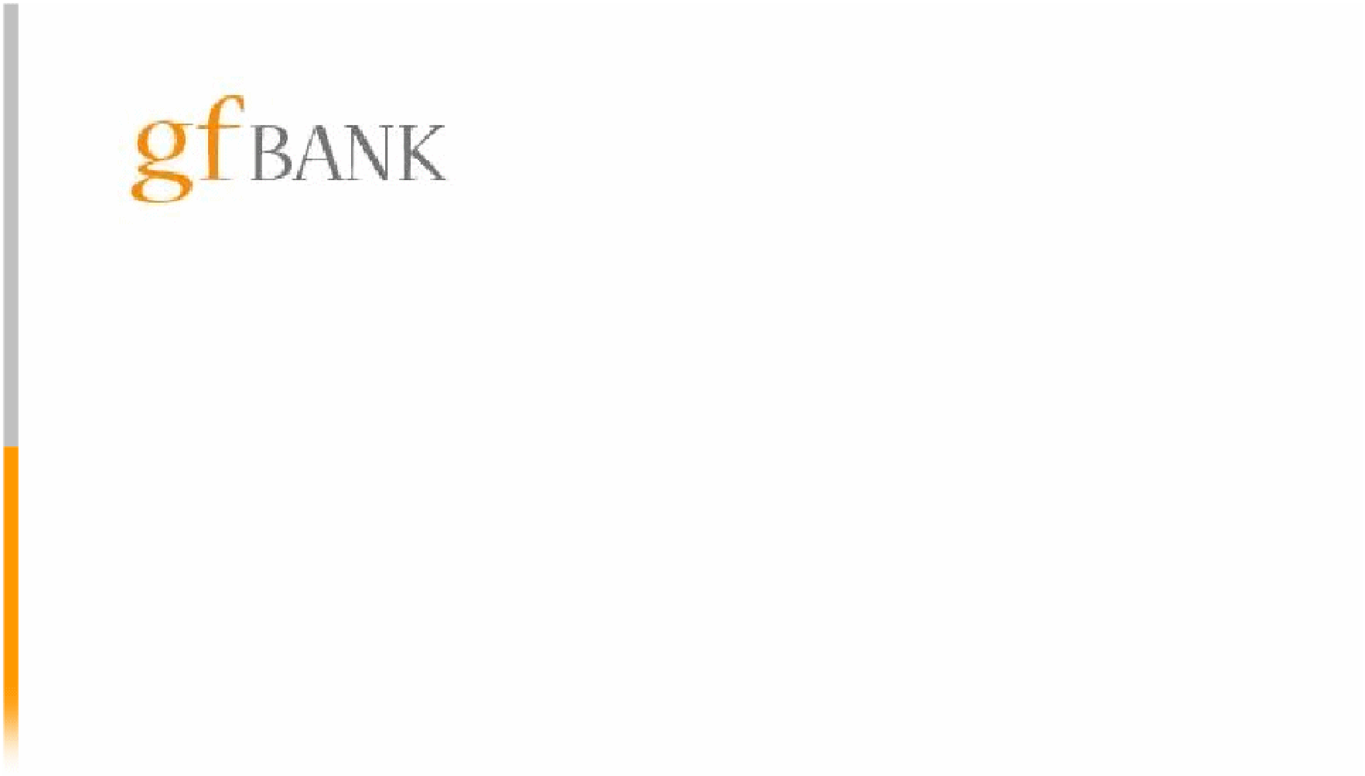


Figure 5.14 An Example of Logo and Template Shapes

The logo is an external image file and the template shapes are drawn using the **Shape** tool ▭.

⊕ Insert the logo that will be used in Layout for the statement's design.

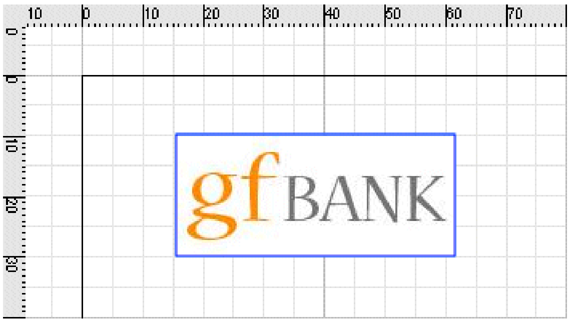
Go to the menu **Insert | Insert Image** and in the Open dialog which opens, browse for the logo image (e.g. Logo.jpg), select it and click on **Open**.

The image is inserted into Layout and appears as an object in the Layout Tree (under the Images family).

⊕ Place the logo on both of the pages.

Select the image from the Layout Tree and drag it into position, using the following screenshot for guidance.

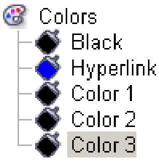
The logo is placed on each page using an Image Area to define its position:



⊕ Consider the colors to be used in your design and create each color that you will use in Layout so that they can be reused effectively throughout your design.

Select the Colors family in the Layout Tree and right-click to reveal the context menu. Select **Insert Color** for each color you need (e.g. light gray, white and orange).

The colors are objects in Layout:



Colors
- Black
- Hyperlink
- Color 1
- Color 2
- Color 3

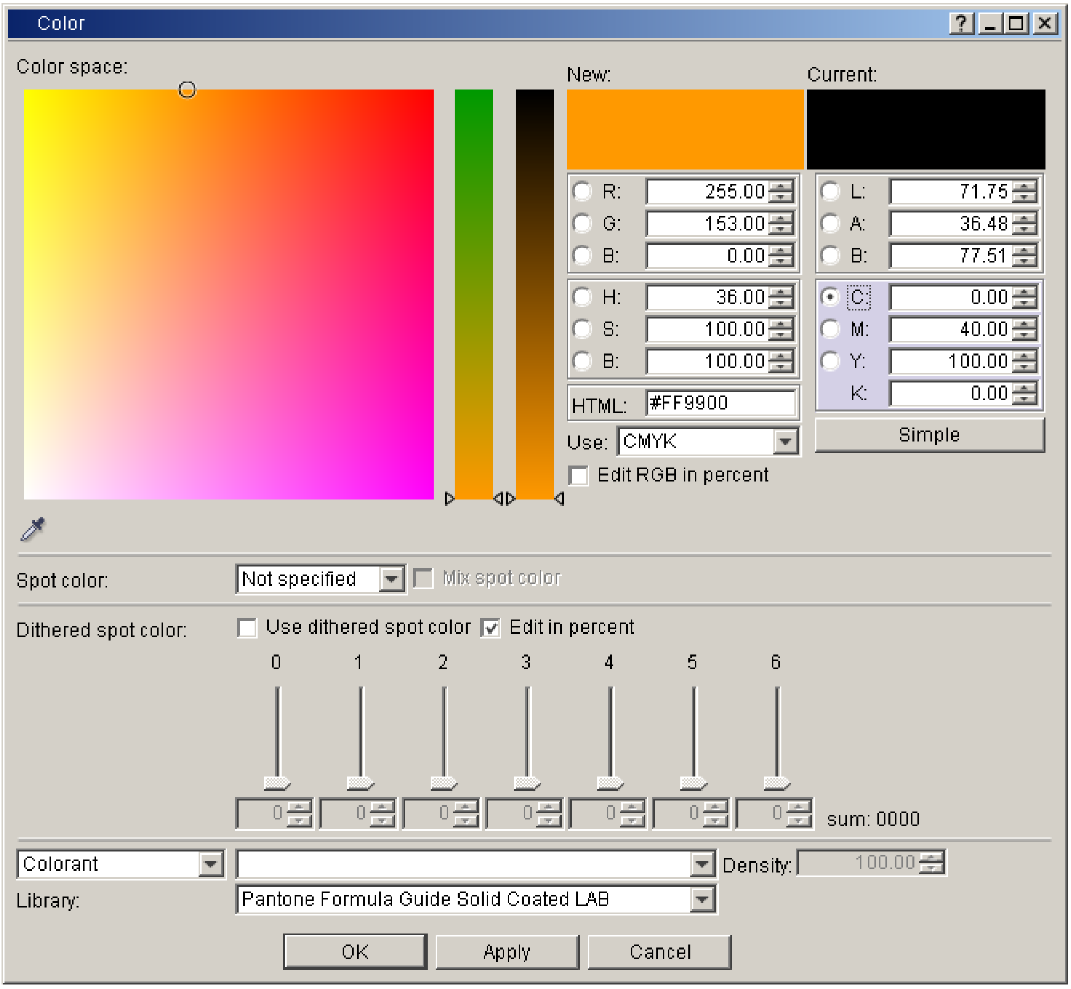
Black and Hyperlink (blue) already exist as colors by default, but in our example we will also be using orange and gray for shapes to enhance the look of the page. We will further be using white as part of a border style.

Remember: When colors are used, they are also created as objects and, like all objects, they should be managed in order to keep the workflow optimized.

⊕ Define one of the colors that you intend to use.

Select a color in the Layout Tree and click on the **Change** button on its properties panel. In the Color dialog that opens select a colorspace in the **Use** combo-box and a shade of color using the map on the left or the edit-boxes on the right, e.g. colorspace *CMYK* and orange).

A color is defined:



Color

Color space:

New: Current:

○ R: 255.00 ○ L: 71.75
○ G: 153.00 ○ A: 36.48
○ B: 0.00 ○ B: 77.51
○ H: 36.00 ● C: 0.00
○ S: 100.00 ○ M: 40.00
○ B: 100.00 ○ Y: 100.00
HTML: #FF9900 K: 0.00
Use: CMYK Simple
□ Edit RGB in percent

Spot color: Not specified □ Mix spot color

Dithered spot color: □ Use dithered spot color ☑ Edit in percent

0 1 2 3 4 5 6
0 0 0 0 0 0 0 sum: 0000

Colorant Density: 100.00

Library: Pantone Formula Guide Solid Coated LAB

OK Apply Cancel

⊕ Confirm the color definition and return to the Layout window.

Click on **OK**.

The Color dialog closes and the color definition is complete.

⊕ Define the other colors that you have created.

Repeat the above two steps for each of the remaining colors, (e.g. light gray and white).

The colors are now created as required.

Remember: Rename the colors suitably (e.g. Orange, LightGray and White).

Remember: Fill styles employ colors so that they can be used to fill other objects.

⊕ Define fill styles that employ the created colors so that those colors can be used in the design.

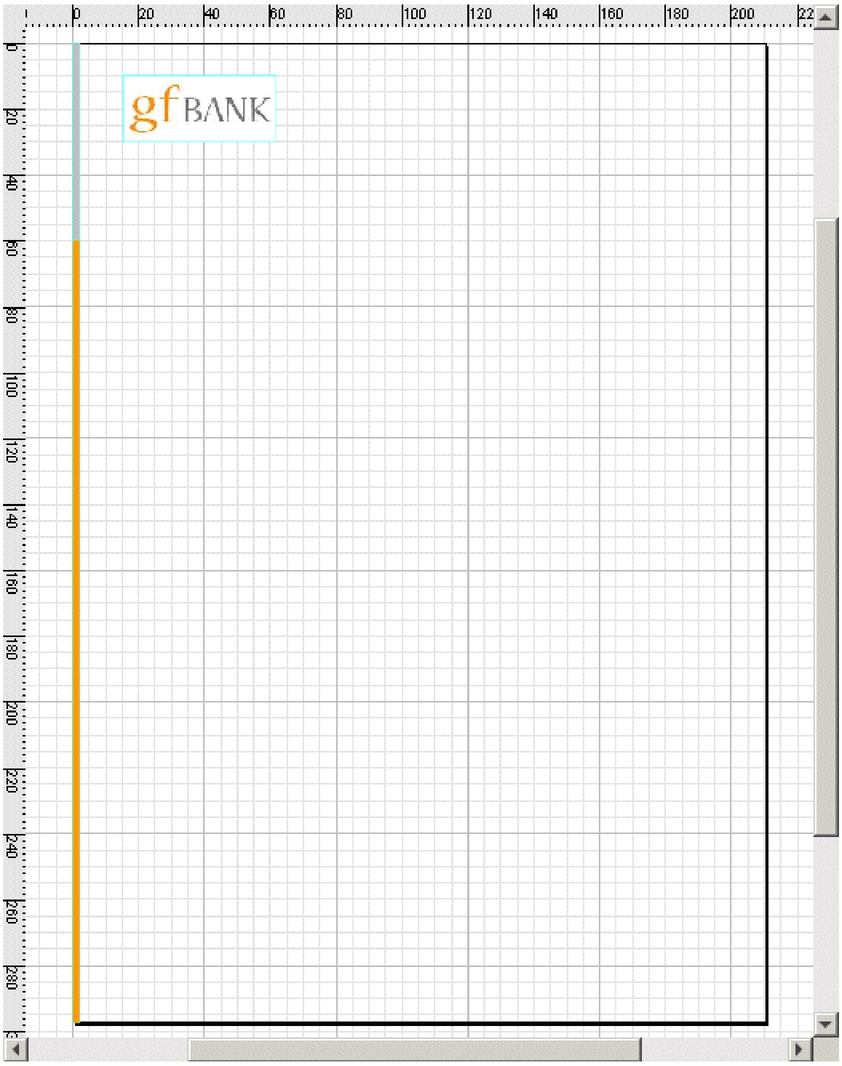
Select the Fill Styles family in the Layout Tree and right-click to reveal the context menu. Select **Insert Fill Style** for each color that you want to use (e.g. light gray, orange and white). Then go to the properties panel of each fill style, select *Color* in the **Type** combo-box and then select the respective color.

Remember: Rename the fill styles suitably (e.g. OrangeFill, LightGrayFill and WhiteFill).

⊕ Create shapes (rectangles) to become part of the template of both pages.

Select the **Shape Tool** (in rectangle mode) ▭ and draw each shape in the Layout Area, using the following screenshot for guidance. Then, on the Content tab of each rectangle's properties panel, select one of the fill styles, that you have previously created, in the **Fill** combo-box (e.g. *LightGrayFill* and *OrangeFill*).

The template shapes are placed on each page:



Shapes can be copied and pasted, should they need to have identical sizes, fill colors etc. in multiple places.

Barcode (3)

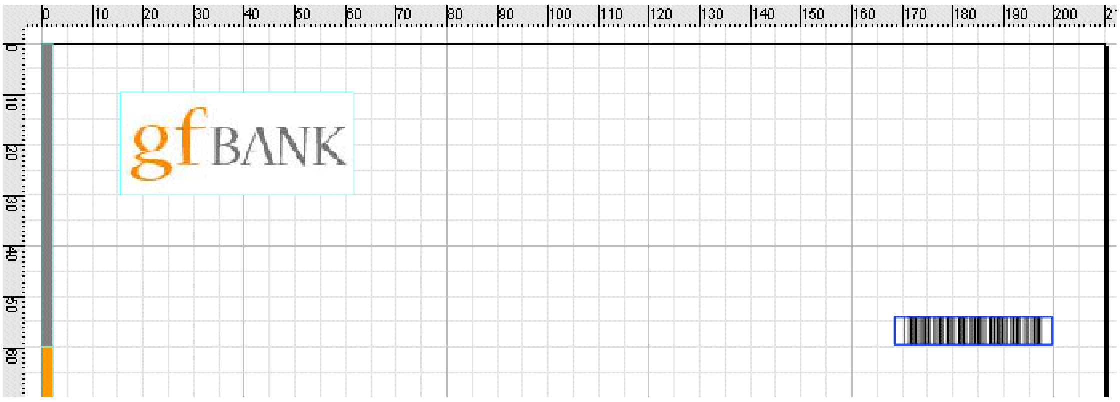
Figure 5.15 An Example Barcode

A barcode is an object that uses variable data to create the barcode's form according to a selected standard. In our example, it will be the individual client's account number.

⊕ Draw a barcode on the first page of the design.

Select the **Barcode Tool** and use the mouse pointer to place it in the Layout Area.

A barcode is placed:

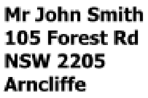


⊕ Set the parameters of barcodes to determine what data will be used to create the barcode.

Select the barcode in the Layout Tree. In its properties panel (on the Content tab) select the account number variable (e.g. Account) from the **Variable** combo-box menu.

The Content tab of the barcode's properties panel also allows you to set other parameters to the barcode. In our example, **Barcode type** has been set to barcode *Code 39*. On the Code 39 tab, the **Height** of the barcode has been changed to *5.5* mm.

Client Address Area (4)



Mr John Smith
105 Forest Rd
NSW 2205
Arncliffe

Figure 5.16 An Example Client Address Area

The Client Address area is a flow area that contains only variable data that is read from the records of the first data input with client data.

In this section we will work with:

1. Client Address Variables
2. Client Address Flow

Client Address Variables

The variables that we will use in our example Client Address Area are:

- Address related variables (e.g. FirstName, Surname, Address, ZIPcode, Town) that are read from the fields of the data input that contains individual clients address details.
- Client gender variable (e.g. Gender) that is converted here in Layout in order to return a standardized greeting conditional on the gender of the client.

  In our example data, the gender field of the Clients.csv file is not filled with text (e.g. Mr or Mrs) but is filled with a simple code string: Either *1* for male; or *2* for female.

⊕ In order to convert the gender variable, go to the variable's properties panel.

Select the variable (e.g. Gender) from the Layout Tree and, in its properties panel, select the Conversion tab.

⊕ Set the conversion type and create conditions for each of the possible values that have been entered in the data field.

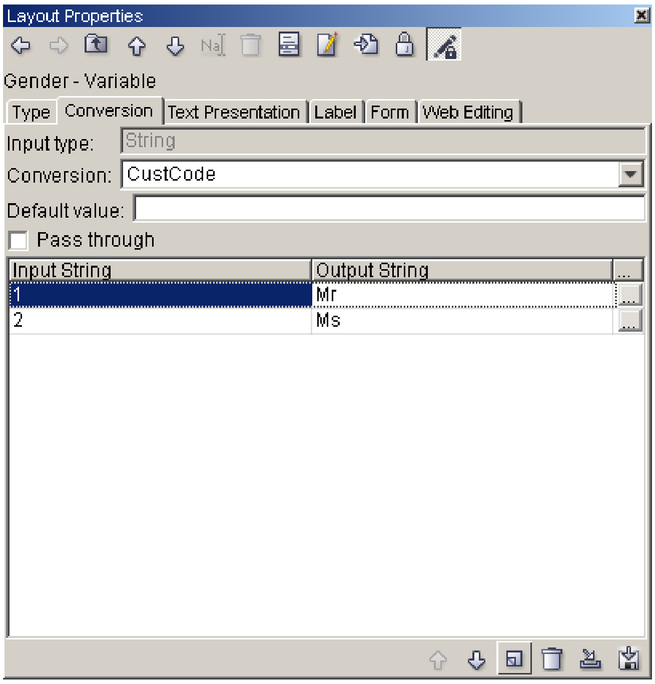
Select *CustCode* from the **Conversion** combo-box menu. Then, click twice on the **Create New** ◫ icon that is available at the bottom of the panel; i.e. once for every possible value (e.g. Gender contains values of either *1* or *2*).

Two conversion rows are added to the conversion area.

⊕ Convert the variable by assigning outputs to each input.

Enter the actual data field values (e.g. *1* and *2*) in the separate rows in the **Input String** column. Then, enter the desired corresponding outputs (e.g. *Mr* and *Ms*) in the **Output String** column.

The conditions of the conversion are set:



Input type: String
Conversion: CustCode
Default value:
☐ Pass through

Input String	Output String
1	Mr
2	Ms

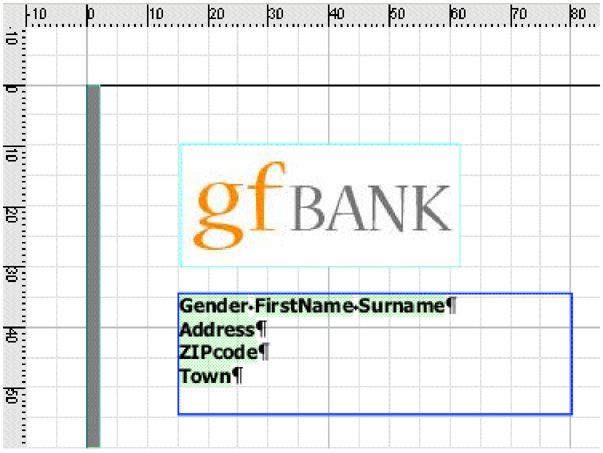
The **Default value** edit-box allows the assignment of an output that should be written in the case that none of the defined conditions are met, i.e. if an entry other than those in the **Input String** column is read from the data field or no entry is found. For our example, the Gender variable will be used in the address field, in combination with FirstName and Surname, so it would be sufficient to leave a blank space (i.e. nothing entered in **Default value**) and the client would still be addressed reasonably well.

Client Address Flow

⊕ Place individual client address details in a position close to the logo, to make a personalized address field on the first page of each statement.

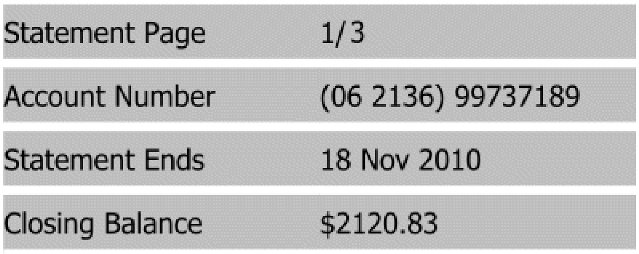
Using the **Flow Area Tool**, draw a flow area on the first page. Then, drag and drop the appropriate variables from the Data family of the Layout Tree to positions within that flow area. Use the following screenshot for guidance.

A flow area containing the clients' addresses is placed as part of the statement:



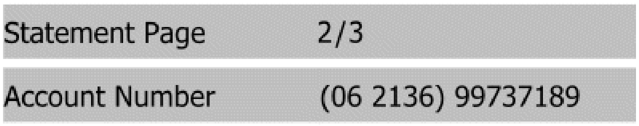
Gender·FirstName·Surname¶
Address¶
ZIPcode¶
Town¶

Statement Details Area (5)



Statement Page	1/3
Account Number	(06 2136) 99737189
Statement Ends	18 Nov 2010
Closing Balance	$2120.83

Figure 5.17 An Example Statement Details Area - On FirstPage



Statement Page	2/3
Account Number	(06 2136) 99737189

Figure 5.18 An Example Statement Details Area - On NextPage

The Statement Details Area is a flow area that contains a table that has two columns. One column contains text stating what the details are and the other contains the details which are variable values.

In this section we will work with:

1. Statement Details Variables
2. Statement Details Table

Statement Details Variables

The variables that we will use in our example Statement Details Area are:

- The Statement Page entry consists of two system variables, GroupSheetCounter and PagesPerRecord, that are presented in the style of "Page X of Y pages".
- The Account Number and Statements Ends entries consist of three record variables (e.g. BranchID, Account Number and EndDate).
- The Closing Balance entry consists of the variable that calculates the balance at the end of the account statement, that we defined in the second data transformer module (e.g. ClosingBalance).

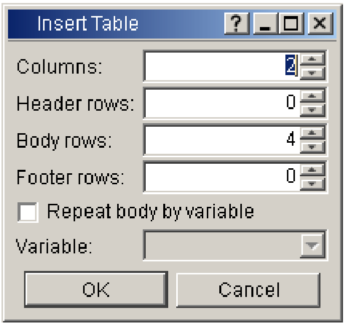
  Remember: It is calculated from the transaction data of each record, i.e the sum of initial balance, all debits and all credits.

Statement Details Table

⊕ Add a table to the top-right of the first page in which to arrange the details and variables that relate to the statement.

Draw another flow area and, keeping the **Flow Area Tool** ▤ active, go to the menu **Table | Insert Table**.

The Insert Table dialog opens.



Insert Table

Columns: 2

Header rows: 0

Body rows: 4

Footer rows: 0

□ Repeat body by variable

Variable:

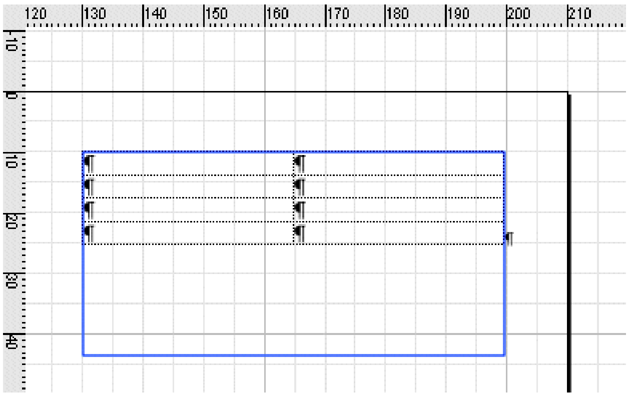
On the first page of our example, we would like to place a page number, the clients account number, a date and the client's account balance in this area (4 sets of details to 4 body rows).

Remember: **Header rows**, **Footer rows** and **Repeat body by variable** are not required for a simple table.

⊕ Define the number of columns and rows that are required for statement details.

Enter the number of columns (e.g. *2*) in the **Columns** edit-box and the required number of rows (e.g. *4*) in the **Body rows** edit-box.

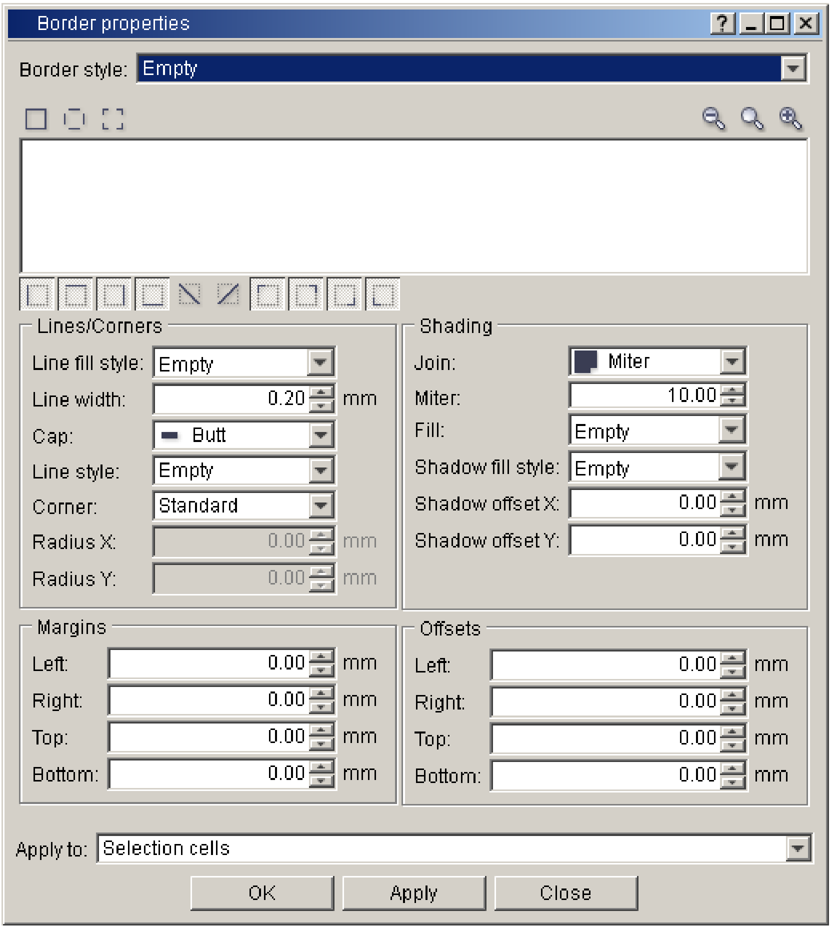
The defined table is nested as part of the flow that is inside the flow area:



⊕ Go to the Border Properties dialog in order to format the style of the table.

Keeping the **Flow Area Tool** active, click in the table and go to the menu **Table | Table Border Style**.

The Border Properties dialog opens:



Border properties

Border style: Empty

Lines/Corners

Line fill style: Empty
Line width: 0.20 mm
Cap: Butt
Line style: Empty
Corner: Standard
Radius X: 0.00 mm
Radius Y: 0.00 mm

Shading

Join: Miter
Miter: 10.00
Fill: Empty
Shadow fill style: Empty
Shadow offset X: 0.00 mm
Shadow offset Y: 0.00 mm

Margins

Left: 0.00 mm
Right: 0.00 mm
Top: 0.00 mm
Bottom: 0.00 mm

Offsets

Left: 0.00 mm
Right: 0.00 mm
Top: 0.00 mm
Bottom: 0.00 mm

Apply to: Selection cells

The Border Properties dialog is for applying border styles. Cell properties are applied on the Cell Properties dialog (go to the menu **Table | Table Cell Properties**). Other table properties, such as the column size and spacing can be found in the table's properties panel. Properties panels are opened by selecting the object in the Layout Tree.

Remember: When creating border styles, it is good practice to employ fill styles that you have previously created.

⊕ Format the table: Add a background fill color and style for the cell borders.

As only the top and bottom borders of the cell should be formatted, it is necessary to select the appropriate icons prior to formatting the table. Make sure only the **Top Line** and **Bottom Line** icons are selected (to be able to select multiple icons press and hold <**Ctrl**> while selecting the icons).

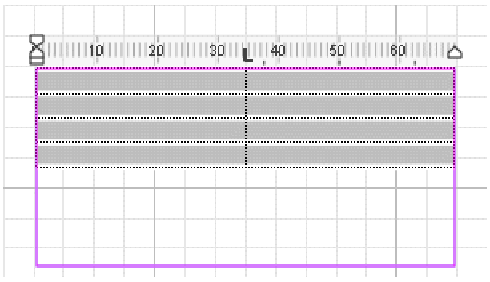
In the Lines/Corners section, choose a fill style that you have created (e.g. *WhiteFill*) from the **Line fill style** combo-box menu and enter a size for the line in the **Line width** edit-box. Then, in the Shading section, select another fill style (e.g. *LightGrayFill*) for the background from the **Fill** combo-box menu.

The properties are previewed as a single cell, in the area at the top of the dialog.

⊕ Apply the border style to all of the cells in the table.

From the **Apply to** combo-box at the bottom of the dialog, select which cells you want the changes to apply to (e.g. *Table cells* means to all of the cells in the table) and click on **OK**.

The table has a border style applied:



Remember: Border styles are also objects and so it is best to rename them (e.g. GrayWhiteBorder), so that they may be easily found and selected later.

⊕ Go to the Table Cell Properties dialog in order to format the cells of the table.

Keeping the **Flow Area Tool** active, click in the table and go to the menu **Table | Table Cell Properties**.

⊕ Format the cells by vertically aligning the text and adjusting the cell height.

In the Cell Properties dialog, select a vertical alignment (e.g. *Center*) from the **Alignment** combo-box menu. Then, select that **Type** should be *Fixed height* and, in the **Fixed height** edit-box that appears, enter a height (e.g. *7 mm*) for the cell.

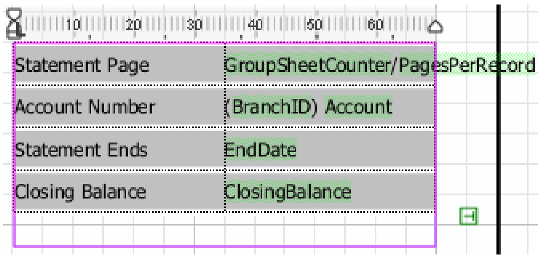
⊕ Apply the cell properties to all the cells of the table.

From the **Apply to** combo-box at the bottom of the dialog, select which cells you want the changes to apply to (e.g. *Table cells*) and click on **OK**. The table is formatted.

⊕ Add statement details variables and related text to the table.

With the **Flow Area Tool** active, drag and drop the previously described record variables, system variables and user defined variables from the Layout Tree to the cells of the right column and type some related text in the corresponding cells of the left column.

The Statement Details Area is complete for the first page:



Statement Page	GroupSheetCounter/PagesPerRecord
Account Number	(BranchID) Account
Statement Ends	EndDate
Closing Balance	ClosingBalance

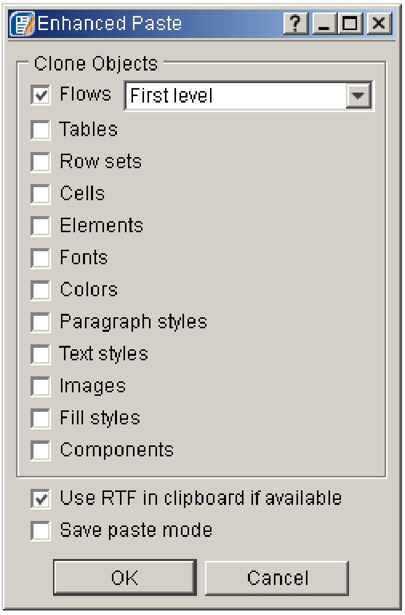
In our example screenshot, it can be seen that the appearance of the Statement Page variables is too big for the flow area, this has caused the **Horizontal Overflow** mark to appear. However, we know that the variable will only return numbers (not a long string like "PagesPerRecord") and so it will fit in the cell when printed. You can proof your work to see that.

On NextPage, only a page number and account number are desired (i.e. 2 sets of details to 2 body rows).

⊕ Create the Statement Details Area for NextPage.

With the **Selection Tool** active select the flow area containing the table with the statement details and copy it (go to the menu **Edit | Copy**). Then go to NextPage, point to a position on the page and perform the enhanced paste command (go to the menu **Edit | Enhanced Paste**).

The Enhanced Paste dialog opens:



Enhanced Paste

Clone Objects

☑ Flows First level
□ Tables
□ Row sets
□ Cells
□ Elements
□ Fonts
□ Colors
□ Paragraph styles
□ Text styles
□ Images
□ Fill styles
□ Components

☑ Use RTF in clipboard if available
□ Save paste mode

Content defining objects can be used in more than one place in Layout. When pasting normally the exact same content object is pasted, i.e. when it is edited it will also be altered in all of the other places that it appears in the design. Therefore, if independent content is required after pasting, it must be cloned. A clone is an object that has the same properties as the copied object when pasted, but is subsequently independent.

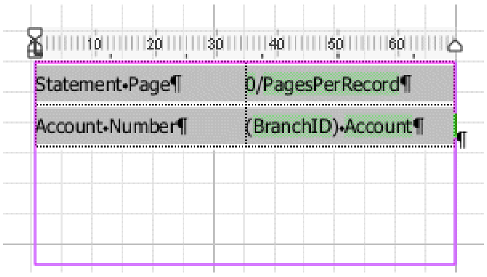
The Enhanced Paste dialog allows the selection of which objects should be cloned as opposed to pasted. In our example, we want the table (and the content of the table) on NextPage to be independent from the table on FirstPage, so we must select that tables should be cloned and that the content within them should also be cloned.

Remember: Tables are built from row sets and cells.

⊕ Define that the tables on FirstPage and on NextPage should be independent of each other.

In the Enhanced Paste dialog, check the **Tables**, **RowSets** and **Cells** check-boxes. Then, click on **OK** and on NextPage, delete the rows that will not be used (e.g. Statement Ends and Closing Balance).

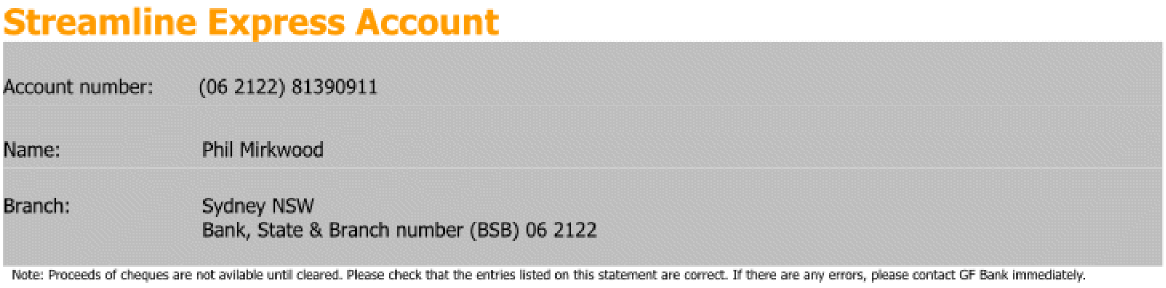
The Statement Details Area is complete for NextPage:



Statement·Page¶	0/PagesPerRecord¶
Account·Number¶	(BranchID)·Account¶

Notice that, when pasted, the flow area has the same name but the content of the flow is given different names. This is because *flow areas* are position defining objects and are, therefore, unique to the page. Because of this they can have the same name on two or more different pages. In contrast, the table is a content defining object and must have a unique name because it can be used on any page. In our example, we have selected that the table should be cloned, consequently the two tables are different objects. This results in having to name the flow and the table with a different name on NextPage than on FirstPage.

Account Details Area (6)



Streamline Express Account

Account number: (06 2122) 81390911

Name: Phil Mirkwood

Branch: Sydney NSW
Bank, State & Branch number (BSB) 06 2122

Note: Proceeds of cheques are not avilable until cleared. Please check that the entries listed on this statement are correct. If there are any errors, please contact GF Bank immediately.

Figure 5.19 An Example Account Details Area

The Account Details Area is a flow area with both simple 'static' text that will appear on every statement and variables that read corresponding client's account details to individual statements.

In this section we will work with:

1. Account Details Variables
2. Account Details Table

Account Details Variables

The variables that we will use in our example Account Details Area are:

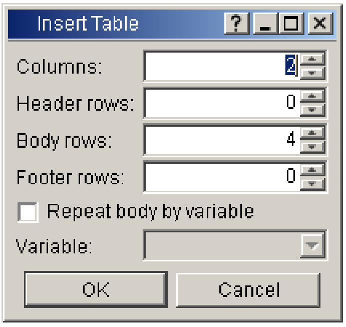
- BranchID and Account – In the "Account number row"
- FirstName and Surname – In the "Name row"
- Location and BranchID – In the "Branch row"

Account Details Table

⊕ Add a table to a position below the Client Address area on the first page, in which to arrange the text and variables.

Draw a flow area and, with the **Flow Area Tool** active, go to menu **Table | Insert Table**.

The Insert Table dialog opens.



Insert Table

Columns: 2

Header rows: 0

Body rows: 4

Footer rows: 0

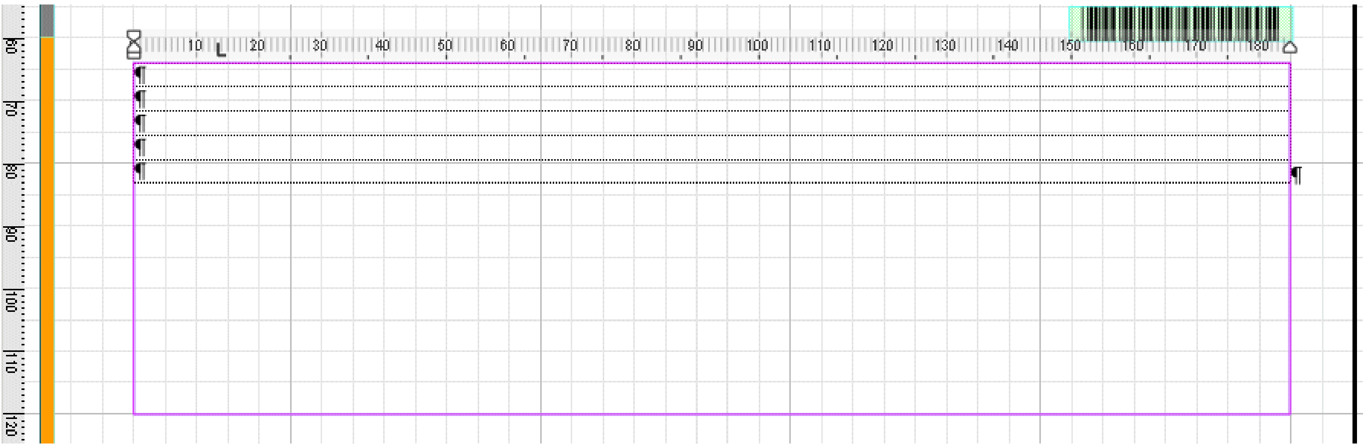
☐ Repeat body by variable

Variable:

⊕ Define the number of columns and rows that are required for account details.

Enter the number of columns (e.g. *1*) in the **Columns** edit-box and the required number of rows (e.g. *5*) in the **Body rows** edit-box.

The defined table is nested as part of the flow that is inside the flow area:



Remember: It can be seen from the paragraph markers in the above screenshot that there are six flows in the flow area. The five flows of the table are nested inside one first-level flow.

⊕ Use the Border Properties dialog to format the style of the table.

Select the middle cells (i.e. the second, third and fourth row) by dragging the mouse pointer (with the **Flow Area Tool** ▦ active) across them. Then, go to the menu **Table | Table Border Style** to open the Border Properties dialog. There you can select the border style that was created for the Statement Details area (e.g. GrayWhiteBorder) for the background from the **Border style** combo-box menu and modify that style (*Modified* appears in the **Border style** combo-box):

- Remove the borders (select *Empty* in the **Line fill style** combo-box).
- Create a **Top** margin for the cells (e.g. *4* mm), so that the text within them will always be aligned equidistant from the cell above.
- To apply the changes made in this dialog select *Selection cells* in the **Apply to** combo-box (which ensures the style is only applied to the cells that were selected) and click on **OK**.

⊕ Use the Cell Properties dialog to format the style of the table.

Select the middle cells (i.e. the second, third and fourth row) by dragging the mouse pointer (with the **Flow Area Tool** ▦ active) across them. Then, go to the menu **Table | Table Cell Properties** to open the Cell Properties dialog. Make the following settings there:

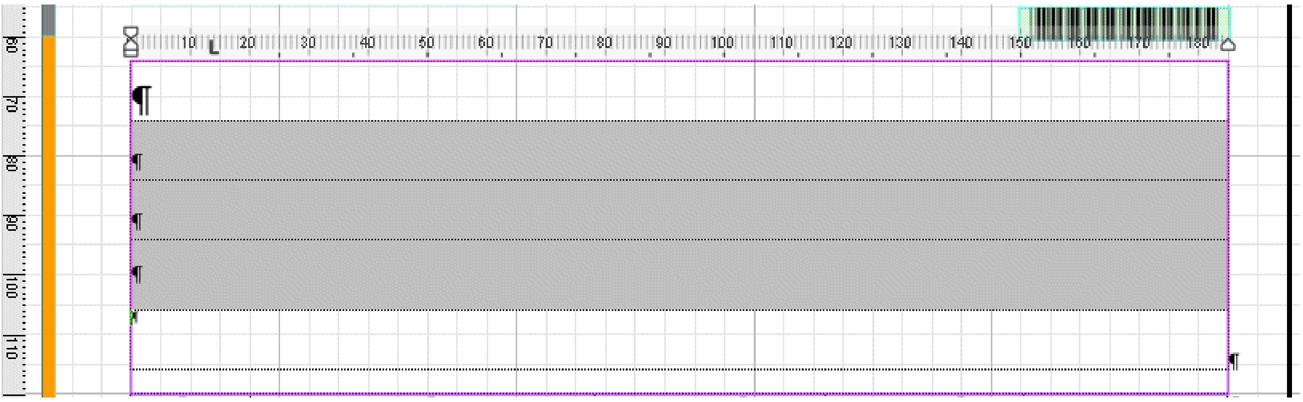
- Select the **Type** as *Custom* and enter a **Min height** (e.g. *10* mm) for the cell.
- To apply the changes made in this dialog select *Selection cells* in the **Apply to** combo-box (which ensures the style is only applied to the cells that were selected) and click on **OK**.

⊕ Format the remainder of the table.

Use the Border Properties and Cell Properties dialogs to finalize your table format:

- Click into the fourth row, open the Border Properties dialog and modify the border style created above by adding a **Bottom** margin (e.g. *4* mm). Make sure *Selection cells* is selected in the **Apply to** combo-box before clicking on **OK**.
- Then click into the unformatted first row, open the Cell Properties dialog and set **Alignment** to *Bottom*. Make sure *Selection cells* is selected in the **Apply to** combo-box before clicking on **OK**.
- Finally, click into the unformatted fifth row, open the Cell Properties dialog and set **Alignment** to *Top*. Make sure *Selection cells* is selected in the **Apply to** combo-box before clicking on **OK**.

The table is formatted:

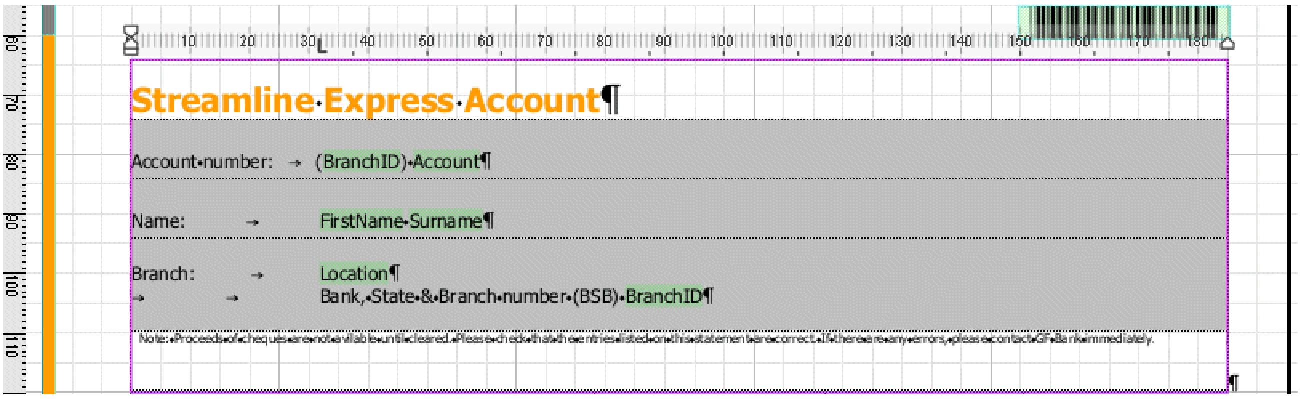


Remember: The Border Properties dialog is for applying border styles to cells.

⊕ Add client's detail variables and related text to the table.

With the **Flow Area Tool** active, type the text like in the screenshot below followed by a tab space and add the variables from the Layout Tree to the cells. Format the text using the Text Format toolbar.

The Account Details Area is complete:



Streamline Express Account

Account number: (BranchID) Account

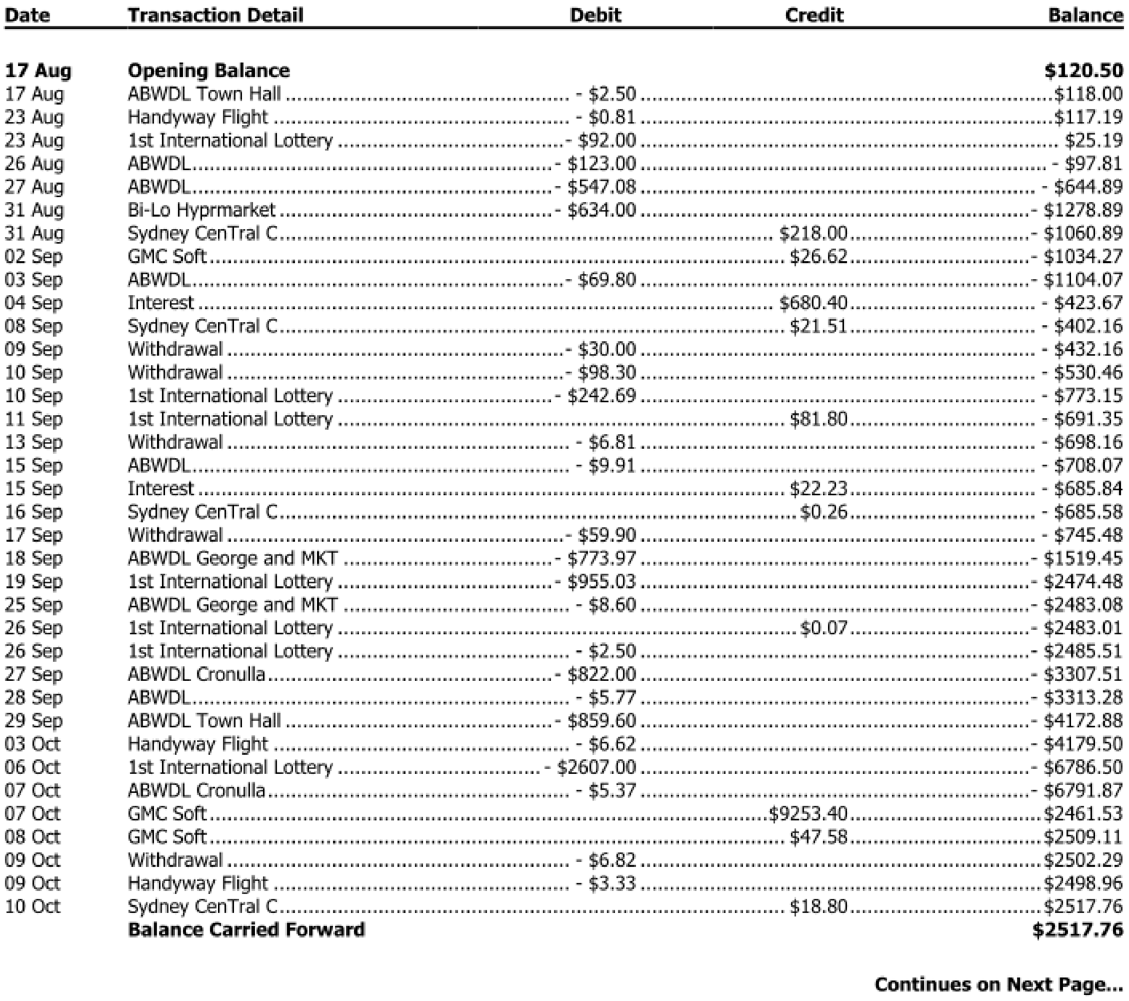
Name: FirstName Surname

Branch: Location
Bank, State & Branch number (BSB) BranchID

Note: Proceeds of cheques are not avilable until cleared. Please check that the entries listed on this statement are correct. If there are any errors, please contact GF Bank immediately.

Remember: When setting text styles you should use fill styles that you have already created. In our example, we have selected the previously created *OrangeFill* for the account name in the first row of this table. This also ensures that the color used for the rectangle shape and for this text will be the same.

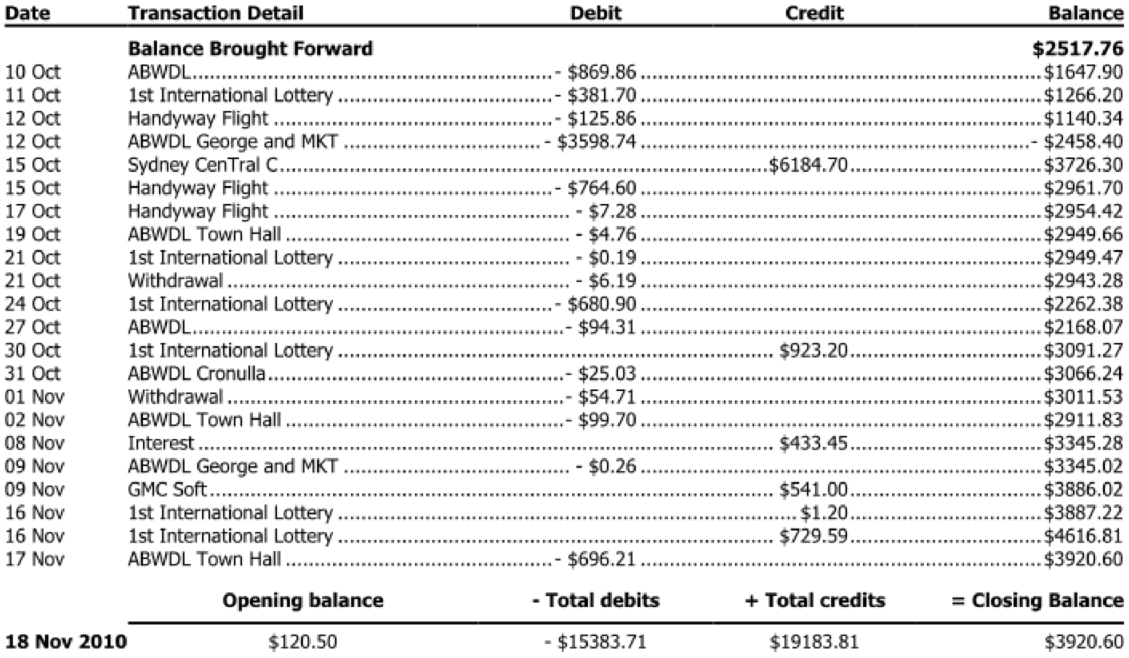
Account Statement Area (7)



Date	Transaction Detail	Debit	Credit	Balance
17 Aug	**Opening Balance**			**$120.50**
17 Aug	ABWDL Town Hall	- $2.50		$118.00
23 Aug	Handyway Flight	- $0.81		$117.19
23 Aug	1st International Lottery	- $92.00		$25.19
26 Aug	ABWDL	- $123.00		- $97.81
27 Aug	ABWDL	- $547.08		- $644.89
31 Aug	Bi-Lo Hyprmarket	- $634.00		- $1278.89
31 Aug	Sydney CenTral C		$218.00	- $1060.89
02 Sep	GMC Soft		$26.62	- $1034.27
03 Sep	ABWDL	- $69.80		- $1104.07
04 Sep	Interest		$680.40	- $423.67
08 Sep	Sydney CenTral C		$21.51	- $402.16
09 Sep	Withdrawal	- $30.00		- $432.16
10 Sep	Withdrawal	- $98.30		- $530.46
10 Sep	1st International Lottery	- $242.69		- $773.15
11 Sep	1st International Lottery		$81.80	- $691.35
13 Sep	Withdrawal	- $6.81		- $698.16
15 Sep	ABWDL	- $9.91		- $708.07
15 Sep	Interest		$22.23	- $685.84
16 Sep	Sydney CenTral C		$0.26	- $685.58
17 Sep	Withdrawal	- $59.90		- $745.48
18 Sep	ABWDL George and MKT	- $773.97		- $1519.45
19 Sep	1st International Lottery	- $955.03		- $2474.48
25 Sep	ABWDL George and MKT	- $8.60		- $2483.08
26 Sep	1st International Lottery		$0.07	- $2483.01
26 Sep	1st International Lottery	- $2.50		- $2485.51
27 Sep	ABWDL Cronulla	- $822.00		- $3307.51
28 Sep	ABWDL	- $5.77		- $3313.28
29 Sep	ABWDL Town Hall	- $859.60		- $4172.88
03 Oct	Handyway Flight	- $6.62		- $4179.50
06 Oct	1st International Lottery	- $2607.00		- $6786.50
07 Oct	ABWDL Cronulla	- $5.37		- $6791.87
07 Oct	GMC Soft		$9253.40	$2461.53
08 Oct	GMC Soft		$47.58	$2509.11
09 Oct	Withdrawal	- $6.82		$2502.29
09 Oct	Handyway Flight	- $3.33		$2498.96
10 Oct	Sydney CenTral C		$18.80	$2517.76
	Balance Carried Forward			**$2517.76**

Continues on Next Page...

Figure 5.20 An Example Account Statement Area - On FirstPage



Date	Transaction Detail	Debit	Credit	Balance
	Balance Brought Forward			**$2517.76**
10 Oct	ABWDL	- $869.86		$1647.90
11 Oct	1st International Lottery	- $381.70		$1266.20
12 Oct	Handyway Flight	- $125.86		$1140.34
12 Oct	ABWDL George and MKT	- $3598.74		- $2458.40
15 Oct	Sydney CenTral C		$6184.70	$3726.30
15 Oct	Handyway Flight	- $764.60		$2961.70
17 Oct	Handyway Flight	- $7.28		$2954.42
19 Oct	ABWDL Town Hall	- $4.76		$2949.66
21 Oct	1st International Lottery	- $0.19		$2949.47
21 Oct	Withdrawal	- $6.19		$2943.28
24 Oct	1st International Lottery	- $680.90		$2262.38
27 Oct	ABWDL	- $94.31		$2168.07
30 Oct	1st International Lottery		$923.20	$3091.27
31 Oct	ABWDL Cronulla	- $25.03		$3066.24
01 Nov	Withdrawal	- $54.71		$3011.53
02 Nov	ABWDL Town Hall	- $99.70		$2911.83
08 Nov	Interest		$433.45	$3345.28
09 Nov	ABWDL George and MKT	- $0.26		$3345.02
09 Nov	GMC Soft		$541.00	$3886.02
16 Nov	1st International Lottery		$1.20	$3887.22
16 Nov	1st International Lottery		$729.59	$4616.81
17 Nov	ABWDL Town Hall	- $696.21		$3920.60

	Opening balance	- Total debits	+ Total credits	= Closing Balance
18 Nov 2010	$120.50	- $15383.71	$19183.81	$3920.60

Figure 5.21 An Example Account Statement Area - On NextPage

The Account Statement area consists of two flow areas (one on FirstPage and one on NextPage) that contain just one flow which contains just one table. The flow areas have a flow order set between them so that if the flow does not have enough room to appear in the first flow area then it will continue to the second. The table is inside that flow.

The table is defined as a repeated table, i.e. the body rows of the table will only appear if there is data to fill them. If there is no more data, the table ends and the flow that contains it ends. Therefore, the data filling the table determines whether it fits into the flow area on FirstPage, or it is necessary to overflow to NextPage.

In addition, the Account Statement table has two different header row sets defined, one that appears if it is displayed on FirstPage and another that appears only on NextPage. Similarly there are two footer row sets defined, that are selected according to whether it is the true end of the table or not.

Remember: Tables can be created with two header and two footer row sets, which is useful for tables which overflow from one page to another.

The content of the table is a mixture of static text and variables arranged to give the typical account statement appearance.

In this section we will work with:

1. Account Statement Variables
2. Account Statement Table
3. Account Statement Overflow

Account Statement Variables

The variables that we will use in our example Statement Details Area are:

- The beginning and end date of the statement (e.g. BeginDate, EndDate) placed in header and footer rows respectively, along with the initial balance and closing balance of the account (e.g. Balance and ClosingBalance).
- Transaction details (e.g. Date, Details, Amount) are repeatedly read into the body rows of the table.
- A *Global Variable* to sum up the transactions as they are read from the data inputs to return a new account balance after each transaction. This will appear along side the details of each transaction and in the headers if the page needs to continue (e.g. "Balance Brought Forward" from the previous page of the account statement).

  Variables are read as they are processed and, therefore, if a variable is a calculation then it is calculated as the page it is on is processed. Global variables can be placed on pages and set-up as calculators that can sum up information that has been processed so far. The result of the calculation can be stored to add to the next global variable that will be processed, as well as being returned as text on the page.

  Keep in mind that a new value is returned by a variable for every repetition (e.g. record, account number, transaction). Thus, it is necessary to link the global variable to another variable and place it close to that variable on the page, i.e. close to that variable within the z-order in the Layout Tree (so that they are processed at the same time). This results in an accumulative total being returned for that variable. In our example, the transaction value variable (e.g. Amount) will be placed in a repeated table to print each new transaction amount to a new row, and so after that position we can use a global variable to return (to the same row) the sum of the transactions that have been processed so far.

  Further, we need to reset the global variable from time to time to avoid the stored total being carried over (e.g. to a different account).

⊕ Create a new variable in order to define a global variable.

Right-click on the Data family of the Layout Tree and select **Insert Variable** from the context menu that appears.

A new variable is added and can be seen in the Layout Tree.

⊕ Define that the new variable will be of a numerical data-type and global kind.

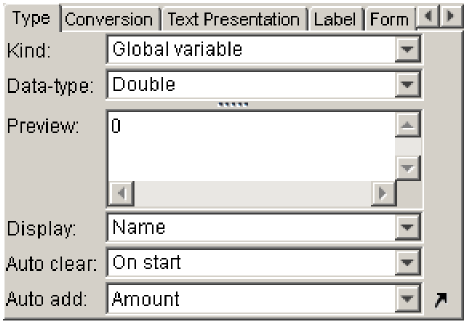
In the variable's properties panel, go to the Type tab and select **Kind** as *Global variable* and **Data-type** as *Double*.

The new variable is a global variable that will use numerical data of double data-type.

⊕ Define for which transaction variable the global variable should return a new sum each time that it is read.

Now select the transaction amount variable (e.g. *Amount*) from the **Auto add** combo-box.

The global variable is defined to add each new transaction amount to the previous global variable calculation:



Kind: Global variable
Data-type: Double
Preview: 0
Display: Name
Auto clear: On start
Auto add: Amount

Remember: Rename the global variable (e.g. BalanceGlobal).

Remember: Global variables store the value they calculate until that stored value is added to the calculation of the next global variable that is processed, creating an accumulative sum. When processing the workflow, global variables will carry their value between pages in the order that they are processed and, if they are not reset, also between records. For our example, we need to reset the global variable at the start of each new account, at which point it should equal the opening balance of the account.

For this, we need to make another new variable that will be placed and processed at the beginning of the table and links the global variable and the opening balance variable.

⊕ Create a new variable on the transactions level, so that a link can be defined between the newly created global variable and the opening balance variable.

Right-click on the 'Value' node located just below the transaction array (e.g. Tr) and select **Insert Variable** from the context menu that appears.

A new variable is added to the Layout Tree.

⊕ Define the kind of variable and add a script that describes that the global variable is assigned the value of the opening balance variable at a certain point, i.e. when used in the Layout.

In the properties panel, select **Kind** as *Calculated*. Go to the Script tab and enter the following script that describes that the global variable is assigned the value of the opening balance variable. Remember to use the location of the other variables in relation to this one:

```
UP6.BalanceGlobal = UP2.Balance;
```

Now a new variable exists that, if it is placed in the design, will 'reset' the value of the global variable to the same value as the opening balance of that account in which it is placed.

"UP" and a number in the script is used in case that you need to reference variables that are a certain amount of levels higher in the data structure. The number, starting from 1, indicates the amount of levels.

Remember: Rename the calculated variable (e.g. InitialBalance).

Account Statement Table

The Account Statement on both pages of our design is one table, but this one table has different headers and footers on each page, depending on whether it is the beginning and end of the table or just a continuation.

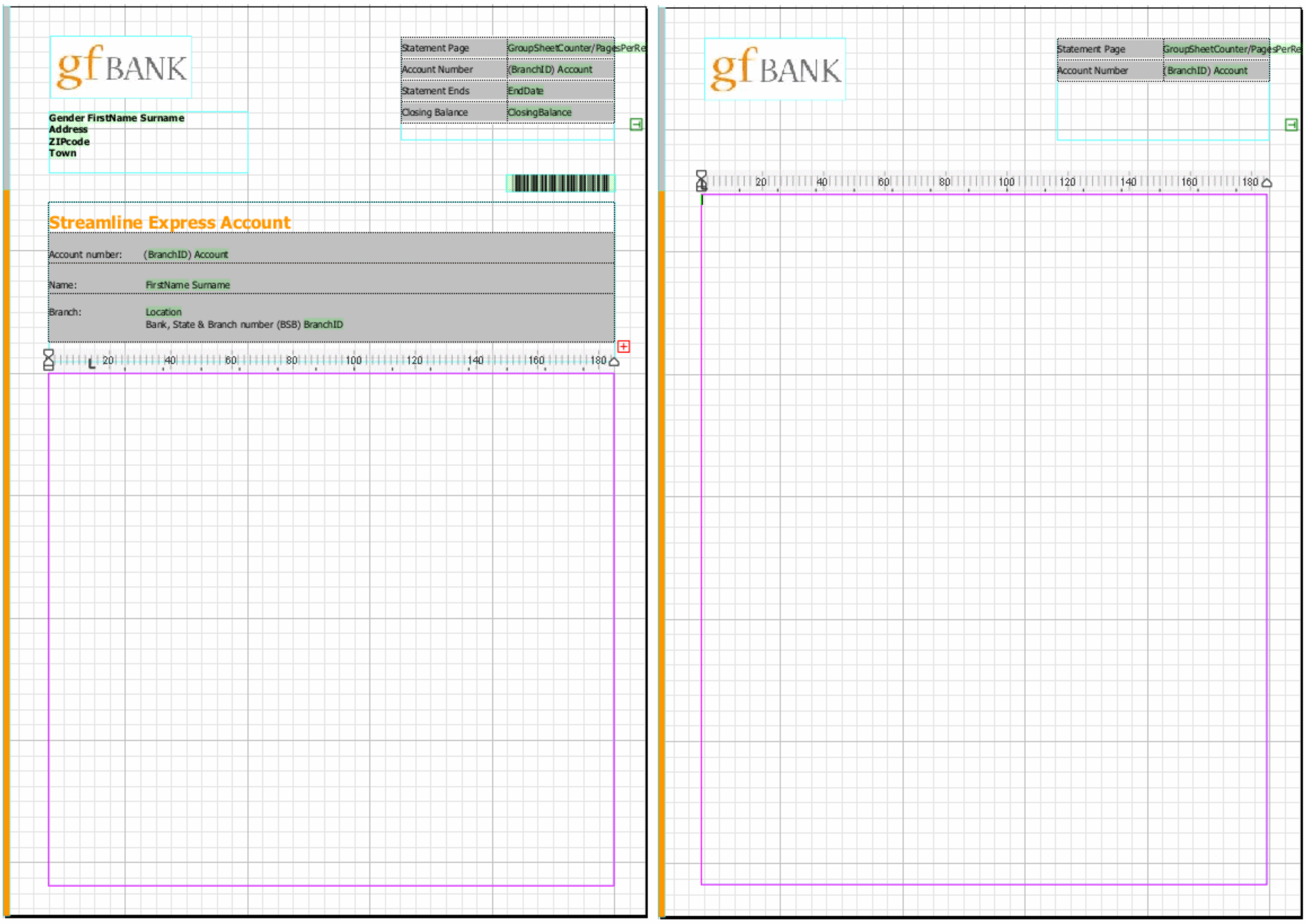
There will be only one body row defined because the body row always contains the same three variables that are read from the transaction data (e.g. Date, Amount and BalanceGlobal). This row must be repeated until there are no more transactions for that account number.

Remember: We can define that the body row should be repeated by the transaction variable (of array *data-type*).

⊕ Add flow areas to both of the pages that will contain the one flow that contains the table.

Using the **Flow Area Tool** ▤ draw large flow areas in the center of both pages.

The two independent flow areas are drawn:

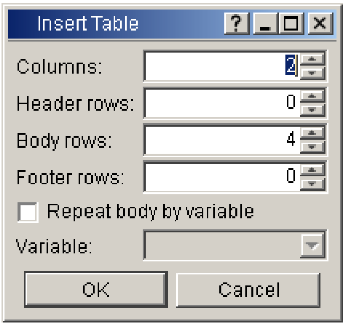


Remember: Rename the flow areas (e.g. AccountStatementFlowArea). It is instructive to name the flow areas at this stage because, soon, they both will be used to contain just one flow (by setting up an overflow from one flow area to the other), containing just one table. Therefore, no more naming will need to be done to objects contained on NextPage because they will be the same objects as are contained on FirstPage.

⊕ Add the table that will be used on both pages of the design.

On the first page, with the **Flow Area Tool** ▤ still active, go to menu **Table | Insert Table**.

The Insert Table dialog opens.



Columns: 2
Header rows: 0
Body rows: 4
Footer rows: 0
☐ Repeat body by variable
Variable:

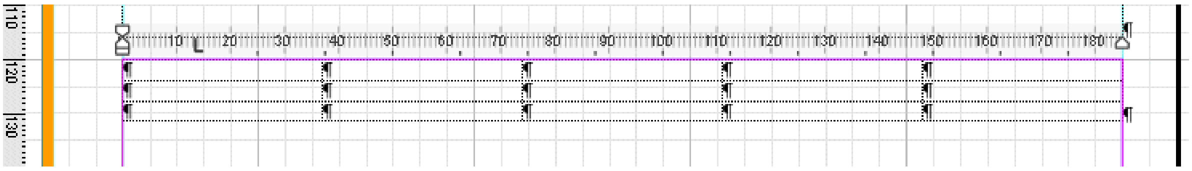
Remember: Tables are made up of row sets. When header, body or footer rows are defined they are kept in separate row sets because they have different functions. For example, if you entered 3 Header rows, 3 Body rows and 3 Footer rows this would result in one header row set, one body row set and one footer row set. These three row sets of *multiple rows* type would each contain three *single rows*.

The nature of row sets seems complex at first and so it is instructive to proceed by inserting the table into the flow area with a minimal amount of rows and building the table in the Layout Tree, i.e. if you want header rows enter *1* header row; if you want body rows enter *1* body row; and if you want footer rows enter *1* footer row.

⊕ Enter the number of columns and the type of rows that are required for the account statement and define that the body rows will be repeated by the transaction data.

Enter the number of columns (e.g. *5*) in the **Columns** edit-box and the required number of rows should be *1* for **Header rows**, *1* for **Body rows** and *1* for **Footer rows**. Then, check the **Repeat body by variable** check-box and from the **Variable** combo-box menu select the transaction array (e.g. *Tr*).

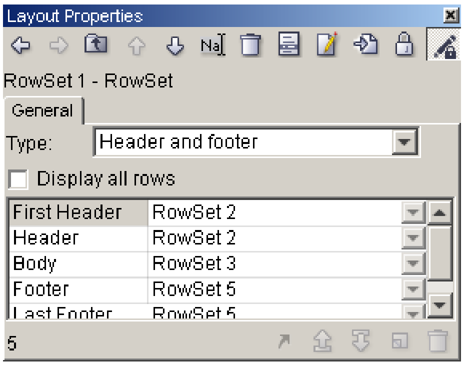
A table with 3 rows is placed in the flow:



Remember: Rename the table (e.g. AccountStatementTable).

Upon expanding the table (and the *header/footer* row set that builds it) in the Layout Tree, the three rows created in the Layout Area can be seen. First, there are the two header row sets (both with the same name and of the type *single row*), then one row set of the type *repeated* and then two footer row sets (both with the same name and of the type *single row*).

Remember: The *header/footer* row set is created by default for every table where at least one header or footer row has been selected, but it is typically used only in cases where it is expected that the table will overflow and it is thus necessary to have two different designs of header or footer. This structure can be recognized and details can be set in the properties panel of the *header/footer* row set:



Layout Properties

RowSet 1 - RowSet

General

Type: Header and footer

☐ Display all rows

First Header	RowSet 2
Header	RowSet 2
Body	RowSet 3
Footer	RowSet 5
Last Footer	RowSet 5

5

Remember: When the *header/footer* row set is selected, you can see that, although there are only three row sets created, two of the row sets (the header and the footer) are entered twice. This is because it is defined that, "At the start of the table the header will be RowSet X and if the table overflows the header will also be RowSet X" and similar with the footer. This structure allows you to easily reassign headers and footers if different ones are needed when a table overflows, i.e. to define the table as: "At the start of the table the header will be RowSet X, but if the table overflows the header will be RowSet Y".

In the properties panel of the *header/footer* row set, this is listed as **First header** and **Header** then **Footer** and **Last footer**.

Remember: Rename the header/footer row set (e.g. AccountStatementRowSet).

In our example, we want to design each of the headers and footers differently, i.e. two first header rows, two header rows, two footer rows and two last footer rows. The different header and footer rows can be designed on the page in the Layout Area, but first we must create independent row sets for the header and footer.

⊕ Assign a new row set to the header, different from the one to be used as first header.

Select the *header/footer* row set in the Layout Tree and in its properties panel select *Create New RowSet* from the drop-down menu in the **Header** row. Then, in the Insert Row Set dialog that opens, select **Type** as *Multiple rows*.

The *single row* type row set gets redefined as a new *multiple rows* type row set.

⊕ Assign a new row set to the footer, different from the one to be used as last footer.

Repeat the previous step for the **Footer**, creating a new row set there.

The row sets of the headers and footers, that will be used when the table overflows, are created.

⊕ Define that the first header of the table will also be a row set of multiple rows.

Select the first (*single row* type) row set inside the *header/footer* row set from the Layout Tree and in its properties panel select *Multiple rows* as **Type**.

The *single row* type row set gets redefined to *multiple rows* type row set.

⊕ Define that the last footer of the table will also be a row set of multiple rows.

Repeat the previous step for the last row set inside the *header/footer* row set.

All *single row* type row sets have been redefined to *multiple rows* type row sets.

Remember: Rename the 4 row sets (e.g. FirstHeader, Header, Footer and LastFooter).

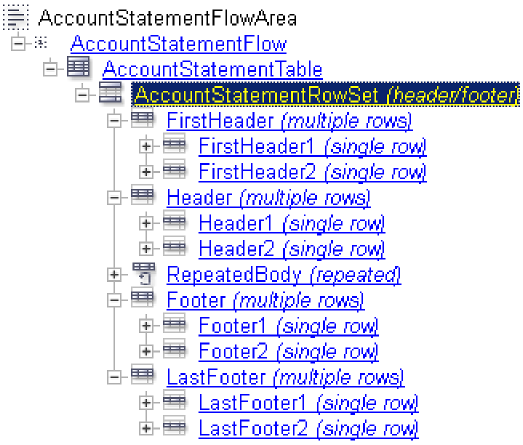
At this point, we have all of the row sets that we need, but the number of rows inside each of them has not been assigned. For example, the header (Header) and footer (Footer) have no rows at all.

⊕ Add the required number of rows to each of the header and footer row sets.

For each of the *multiple rows* type row sets inside the *header/footer* type row set, go to its properties panel and click on the **Create New** ▣ icon. Then, in the Insert Row Set dialog that opens, select *Single row* for **Type**.

Repeat this step until each of the *multiple rows* type row sets (e.g. FirstHeader, Header, Footer and LastFooter) have the required number of rows (e.g. 2). The new rows are part of the row sets.

Remember: Rename the 8 row sets (e.g. FirstHeader1, FirstHeader2, Header1 etc).



```
AccountStatementFlowArea
  AccountStatementFlow
    AccountStatementTable
      AccountStatementRowSet (header/footer)
        FirstHeader (multiple rows)
          FirstHeader1 (single row)
          FirstHeader2 (single row)
        Header (multiple rows)
          Header1 (single row)
          Header2 (single row)
        RepeatedBody (repeated)
        Footer (multiple rows)
          Footer1 (single row)
          Footer2 (single row)
        LastFooter (multiple rows)
          LastFooter1 (single row)
          LastFooter2 (single row)
```

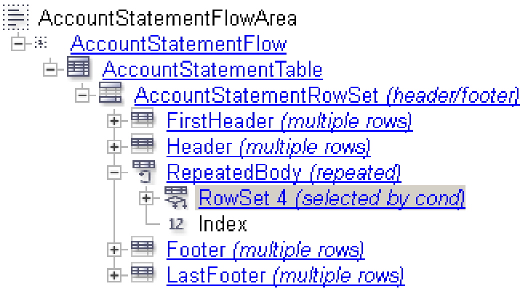
Next we will look at the repeated body row. In this example, we would like to have two different rows here as well: One for the debits that shows the transaction value (e.g. Amount) in the Debit column; and one for the credits that shows the transaction amount in the Credits column. This requires a slightly different approach as we need the repeated row set to remain of type *repeated* but to define that the single row, that it contains, should be selected from two possible rows depending on a condition.

Remember [115]: The repeated body is repeated by the transaction array, i.e. the row set will be called to print for each transaction that is read from the data. Therefore, we need to print either: one row (if the transaction value is a debit); or the other row (if the transaction value is a credit). Thus, the condition for body row selection will be based on that logic.

⊕ Define that the repeated row set selects the row which is displayed according to some condition.

Expand the *repeated* row set (e.g. RepeatedBody) in the Layout Tree to reveal the *single row* type row set that it contains. Select that row and, in its properties panel, select *Select by condition* in the **Type** combo-box.

The *single row* type row set gets redefined as a conditional row (*selected by cond*).

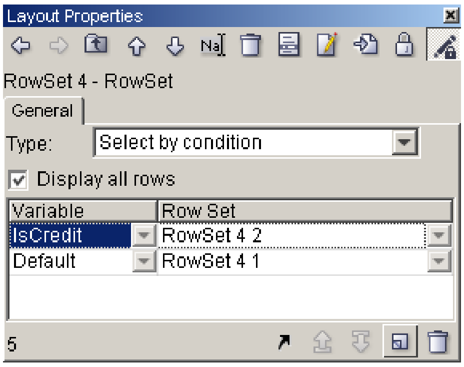


```
AccountStatementFlowArea
  AccountStatementFlow
    AccountStatementTable
      AccountStatementRowSet (header/footer)
        FirstHeader (multiple rows)
        Header (multiple rows)
        RepeatedBody (repeated)
          RowSet 4 (selected by cond)
          12 Index
        Footer (multiple rows)
        LastFooter (multiple rows)
```

Remember: A Boolean variable (e.g. IsCredit) was created in the first Data Transformer, to return true if the transaction value is positive. Now, we will use that Boolean variable to select an alternative row (which will later be given an alternative style).

⊕ Add an alternative row to the repeated row set and assign it a conditional variable, so that if that condition is met then that alternative row will be printed.

In the properties panel of the *select by cond* row set, click on the **Create New** icon and, in the Insert Row Set dialog that opens, select **Type** as *Single row*. Then, in the new row that appears on the General tab, select the Boolean variable that has been created to indicate that the transaction amount is positive (e.g. IsCredit):



RowSet 4 - RowSet

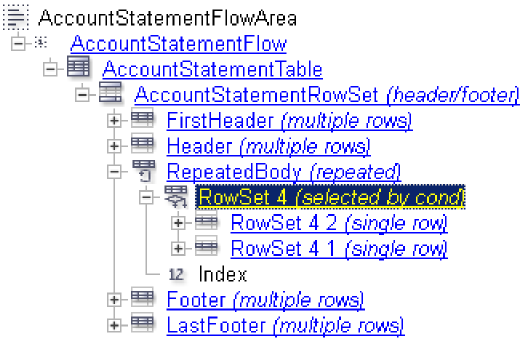
Type: Select by condition

☑ Display all rows

Variable	Row Set
IsCredit	RowSet 4 2
Default	RowSet 4 1

5

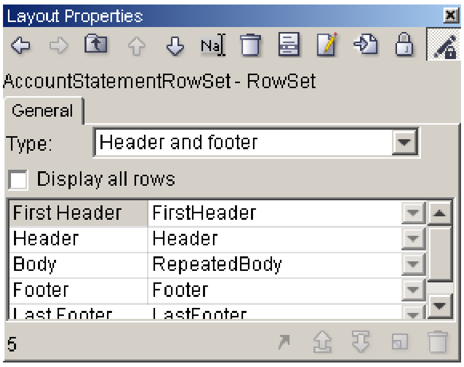
The repeated row set now contains a conditional row set that contains a default row and an alternative row that will replace it if the condition is met (e.g. IsCredit returns *True*):



```
AccountStatementFlowArea
  AccountStatementFlow
    AccountStatementTable
      AccountStatementRowSet (header/footer)
        FirstHeader (multiple rows)
        Header (multiple rows)
        RepeatedBody (repeated)
          RowSet 4 (selected by cond)
            RowSet 4 2 (single row)
            RowSet 4 1 (single row)
          12 Index
        Footer (multiple rows)
        LastFooter (multiple rows)
```

Remember: Rename the row sets (e.g. IsCredit and IsDebit).

We now have all of the rows that we need for the table (e.g. two first header rows, two header rows, one repeated body row, two footer rows and two last footer rows) and we are sure that each row is grouped correctly in row sets, according to its function:



AccountStatementRowSet - RowSet

Type: Header and footer

□ Display all rows

First Header	FirstHeader
Header	Header
Body	RepeatedBody
Footer	Footer
Last Footer	LastFooter

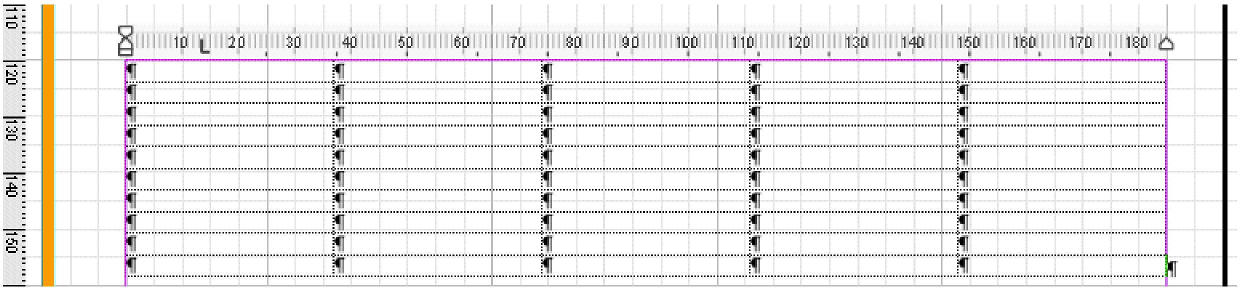
5

By default, not all of these new rows will be visible on the page (e.g. FirstPage) in the Layout Area, as only one variant of any conditional row can be seen. Therefore, only the correct number of rows, that will be seen when the workflow is printed, can be seen now. However, we need to apply content to all of the rows so it would be helpful to see all of the rows on the page. This can be done with the **Display all rows** check-box.

⊕ Make all of the defined rows visible in the Layout Area so that they can be edited and formatted.

Check the **Display all rows** check-box in the properties panel for the *header/footer* type row set.

All of the new rows are visible on the first page of the layout design.



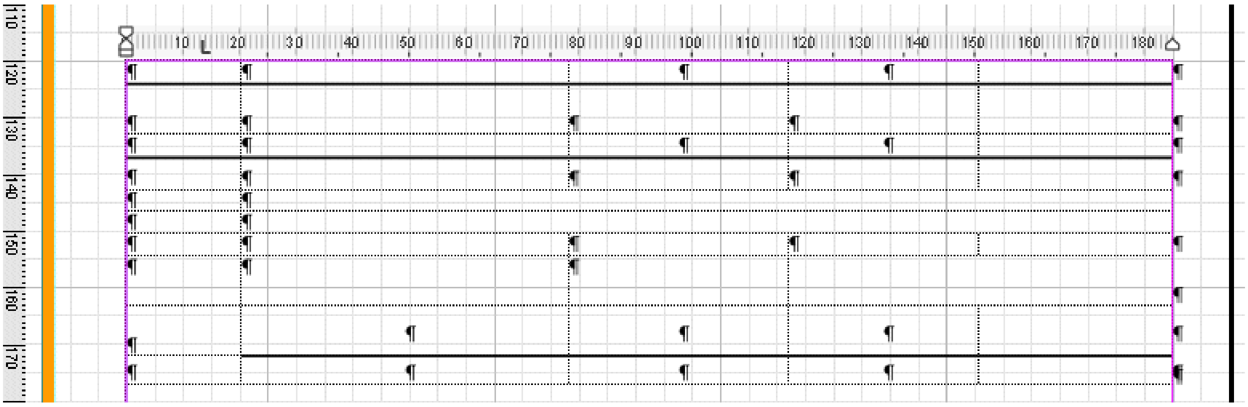
The rows are displayed in the order that they appear in the Layout Tree. In our example, this means that the first two rows are the first header rows and the second two rows (third and fourth) are the header, then the next two are the repeated rows (e.g. fifth row = IsCredit) etc. Bear this order in mind when designing your table.

⊕ Create and apply border styles to finalize your table format.

To apply border styles proceed by the following steps:

- Select the BorderStyles family and right-click on it and select **Insert Border Style**. Make the following settings for this newly created border style:
    - Select the appropriate icon(s) so that the settings related to lines are applied only to the parts of the border you want to change. In this case this is the bottom border of the cell, so select the **Bottom Line** ▭ icon.
    - In the **Line fill style** combo-box select *BlackFill.*
    - Set the **Line width** to *0.50* mm.
    - On the Margins tab set *0.30* mm in the **Bottom** combo-box and *3* mm in the **Top** combo-box.
    - Now rename this border style appropriately (e.g. UnderlineStyle).
- Select all of the cells of the first row and open the Border Properties dialog (opened from the menu **Table | Table Border Style**) and select the newly created border style (e.g. *UnderlineStyle*) in the **Border Style** combo-box at the top of the dialog. Confirm the dialog; the created border style will be applied to the cells which were selected.
- Repeat this last step for the third row of the table.

- To create the different heights of rows you can see e.g. in the second row of the screenshot below, create a new border style and set the value in the **Top** combo-box on the Margins tab to *5* mm. Then select this row and apply this new border style to the row using the Border Properties dialog in the same way as described above.
- Combine the procedures described above to create and apply further border styles to rows and cells according to the demands of the design you are creating.

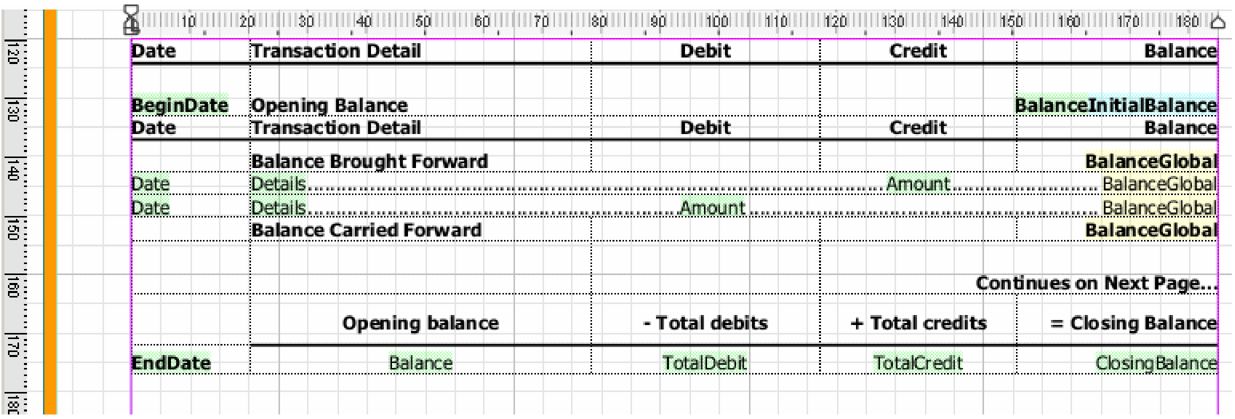


Remember: The Border Properties and Cell Properties dialogs are for applying border styles and other cell properties.

Some cells have been merged using the **Span left** or **Span right** properties of cells (alternatively, it is possible to merge cells by selecting them and then selecting **Merge Cells** from the right-click menu).

Add statement and transaction detail variables and related text to the table and use paragraph styles to control their appearance in some cells.

With the **Flow Area Tool** active, type some relevant text and drag and drop variables from the Layout Tree to the cells of the account statement table on the first page of the design. Format the text using the Text Format toolbar. Use the following screenshot for guidance:

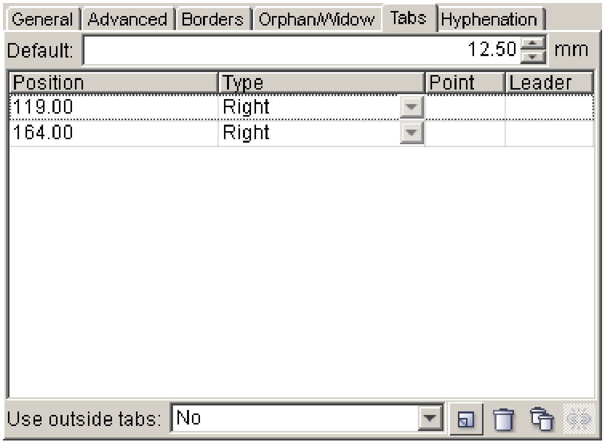


Date	Transaction Detail	Debit	Credit	Balance
BeginDate	Opening Balance			BalanceInitialBalance
Date	Transaction Detail	Debit	Credit	Balance
	Balance Brought Forward			BalanceGlobal
Date	Details..Amount........................BalanceGlobal			
Date	Details....................................Amount....................................BalanceGlobal			
	Balance Carried Forward			BalanceGlobal
			Continues on Next Page...	
	Opening balance	- Total debits	+ Total credits	= Closing Balance
EndDate	Balance	TotalDebit	TotalCredit	ClosingBalance

In this example (in the sixth and seventh rows), a special paragraph style has been set for the body rows. In both of these rows, three variables have been placed in a large merged cell. The placement of these three variables within the cell is controlled using tabulators defined in paragraph styles. Additionally, leaders (i.e. the .... effect between the Details, Amount and BalanceGlobal variables) are employed to fill in spaces that are controlled by the tabulators. To create this kind of paragraph style perform the following steps:

- Select the ParagraphStyles family, right-click on it and select **Insert Paragraph Style** to create a new paragraph style to be used to control the appearance of the text in the sixth row. Rename it appropriately.
- Go to the Tabs tab of the properties panel and click on the **Create New** icon twice to create two rows where you can set tabulators and leaders. We need two tabulators in order to be able to define the two spaces between the three variables.

- Determine the **Position** of the tabulators, their **Type** (i.e. how the text will be positioned in relation to the tabulator) and the **Leader** to be used:



Default: 12.50 mm

Position	Type	Point	Leader
119.00	Right		
164.00	Right		

Use outside tabs: No

- Repeat the steps described above for a new paragraph style that will be applied to the seventh row; only the positions of the tabulators have to be modified accordingly.

- Select the according rows and apply the appropriate newly created paragraph styles by selecting them from the ParaStyle toolbar located at the top of the window.

The formatting of numerical variables will be explained in the Data Format section of this example.

The design of the Account Statement Area is complete.

Remember [26]: Note the position of the global variable that has been created to calculate the running total of the transactions (e.g BalanceGlobal) and its essential partner variable that resets it to the starting balance for each account (e.g. InitialBalance). Their positions on the page in the above screenshot of the page layout (e.g. Initial Balance in FirstHeader and BalanceGlobal in Header, RepeatedBody and Footer) are important to ensure the correct functionality.

Account Statement Overflow

Finally for the Account Statement Area, we must set up the overflow for the account statement flow area on FirstPage. Defining that the account statement flow (containing the account statement table) will continue to the account statement flow area on NextPage.

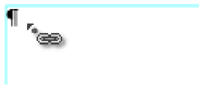
Remember: We have already set up the page order when the pages were defined, therefore we only need to select the flow area to which the continuing table can flow.

Remember: Overflows can be defined using the **Flow Order Tool** .

⊕ Determine that the account statement area on FirstPage will overflow to the account statement area on NextPage.

Select the **Flow Order Tool** and click once in the account statement flow area on FirstPage. Then, navigate to NextPage using the Layout Tree and hover the mouse pointer above the account statement flow area there.

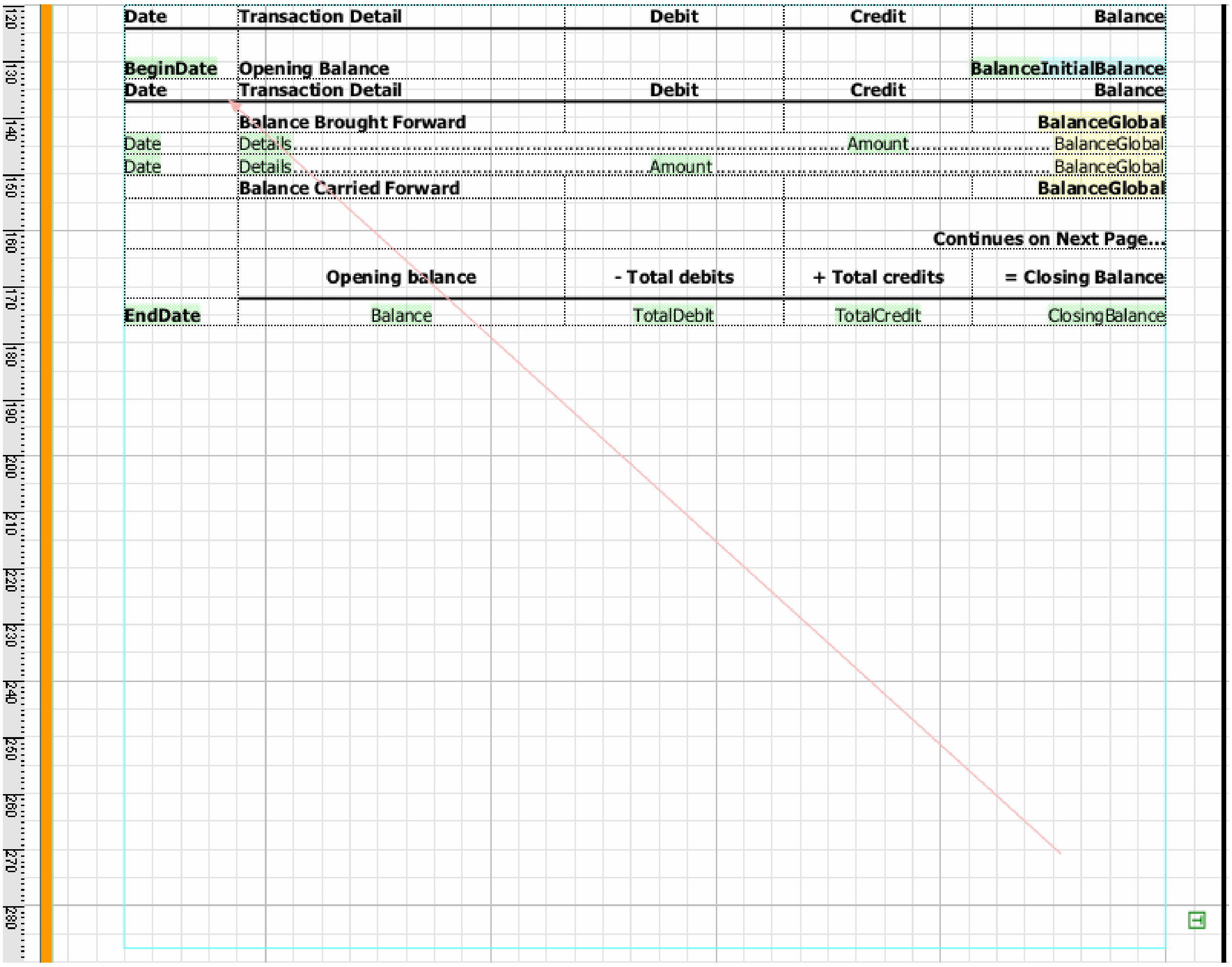
A flow order mouse pointer is shown over the flow area:



⊕ Determine that the account statement area on NextPage will receive the flow from FirstPage should it overflow.

Click once in the account statement flow area on NextPage.

The **Flow to Next Page** marker (pink arrow) is shown over the flow area and the content of the flow of FirstPage (i.e. the table) is displayed:

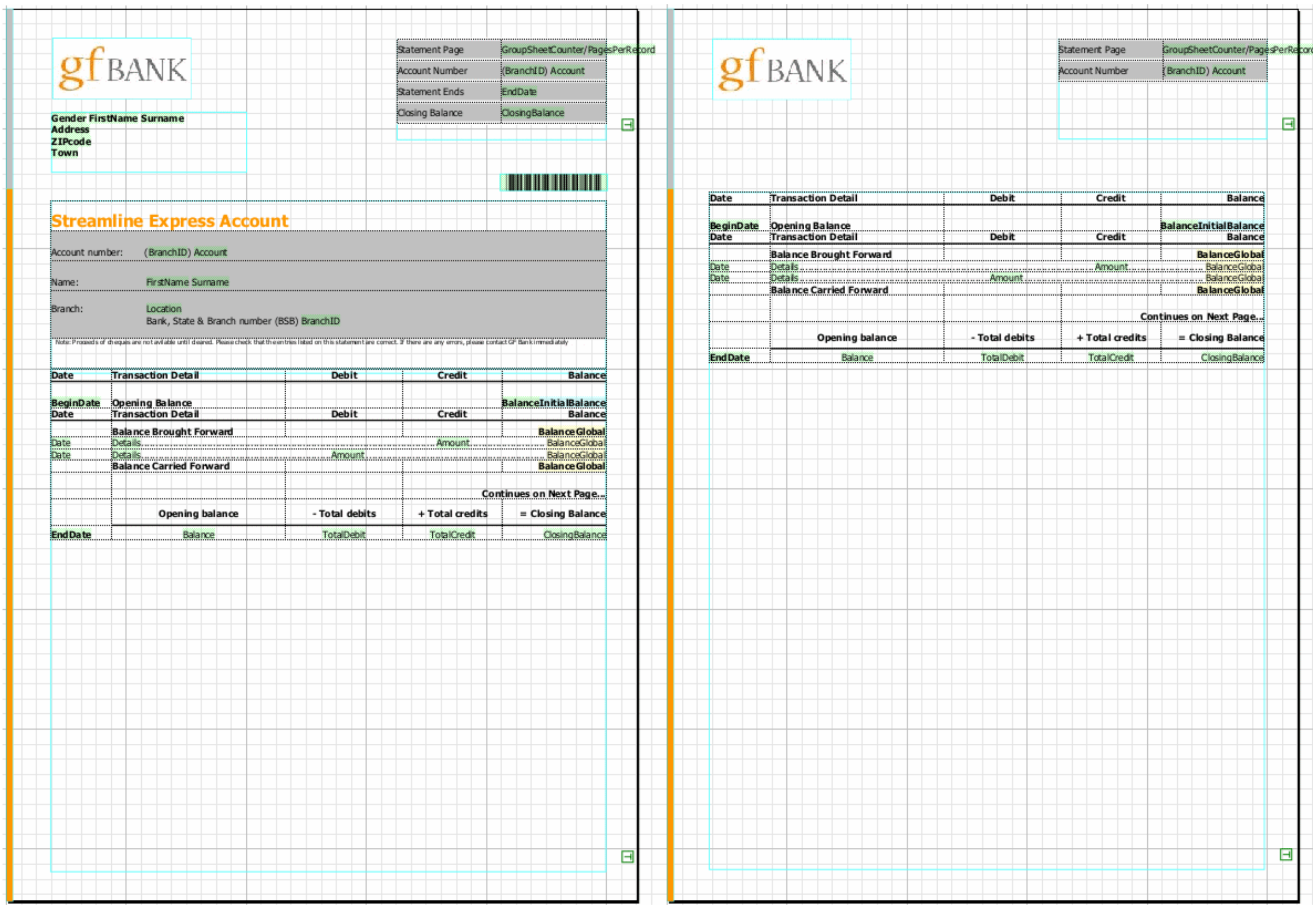


Date	Transaction Detail	Debit	Credit	Balance
BeginDate	Opening Balance			BalanceInitialBalance
Date	Transaction Detail	Debit	Credit	Balance
	Balance Brought Forward			BalanceGlobal
Date	Details		Amount	BalanceGlobal
Date	Details	Amount		BalanceGlobal
	Balance Carried Forward			BalanceGlobal
				Continues on Next Page...
	Opening balance	- Total debits	+ Total credits	= Closing Balance
EndDate	Balance	TotalDebit	TotalCredit	ClosingBalance

The **Flow to Next Page** marker (pink arrow) will also be shown over the account statement flow area on FirstPage.

You can check to see that the correct overflowing behavior has been assigned on the Content tab of the flow area Layout Properties. The account statement flow area on both pages should have the account statement flow selected as **Content** and the option **Flowing to next page** checked.

The overflowing Account Statement Area is complete which completes the design of the Layout pages:



Data Format (8)

If you were to proof the workflow now, the text format of the placed variables throughout the pages of the design would be wrong for the needs of our example.

Remember: We transformed the *data-types* in the first Data Transformer module so that we could manipulate them properly. Now we no longer need to manipulate them and would just like them to be presented properly on the page, i.e. format them. This can be done using the Conversion tab of each variable.

⊕ For each of the variables of DateTime data-type that are placed in the design, format the returned values.

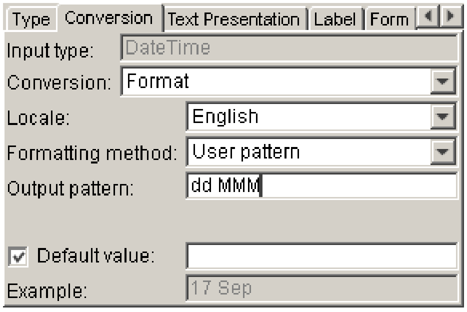
Select each variable (e.g. Date) from the Layout Tree and, on the Conversion tab of its properties panel, select **Conversion** as *Format*.

The tab changes to reveal other formatting options.

⊕ Define a formatting method that will return just Day and Month.

Select *User pattern* in the **Formatting method** combo-box and, in the Output Pattern combo-box that appears, enter a pattern using the code letters d for day and M for Month (e.g. *dd MMM*).

The date is formatted how it will be printed:



Input type: DateTime

Conversion: Format

Locale: English

Formatting method: User pattern

Output pattern: dd MMM

☑ Default value:

Example: 17 Sep

The **Example** field at the bottom of the tab allows a preview of how the date will be presented when printed, allowing you to experiment with different formats. In our example, we used a different format for the variable that returns the end of statement date (e.g. EndDate) so that it includes the year also, i.e. *dd MMM yyyy* (e.g. 28 Nov 2006).

⊕ For each of the variables that represent monetary values and are actually placed in the design, format the returned values.

Select each variable (e.g. Amount, Balance, BalanceGlobal, ClosingBalance, TotalDebit and TotalCredit) from the Layout Tree and, on the Conversion tab of its properties panel, select *BC.-String* in the **Conversion** combo-box.

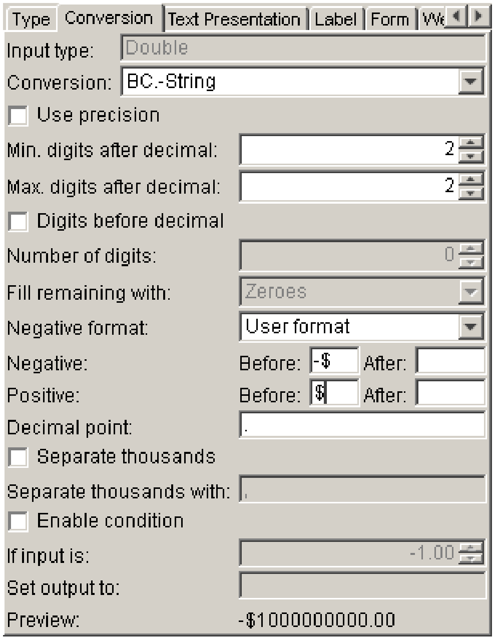
The tab changes to reveal other formatting options.

We are converting the numerical data-type (double) back to string data-type because basic text characters provide the most flexible formatting.

⊕ Define a format that will return all values to two decimal places and place currency symbols in front of those values.

Uncheck the **Use precision** check box and, in the **Min. digits after decimal** and **Max. digits after decimal** combo-boxes that appear, enter the required amount of decimal places (e.g. *2*). Then, in the **Negative format** combo-box, select *User format* and, in the **Negative** and **Positive** edit boxes, enter the text characters that are required to appear depending on whether the value is negative or positive (e.g. *- $* and *$*).

Monetary values are formatted how they will be printed:



Input type: Double

Conversion: BC.-String

□ Use precision

Min. digits after decimal: 2

Max. digits after decimal: 2

□ Digits before decimal

Number of digits: 0

Fill remaining with: Zeroes

Negative format: User format

Negative: Before: -$ After:

Positive: Before: $ After:

Decimal point: .

□ Separate thousands

Separate thousands with: ,

□ Enable condition

If input is: -1.00

Set output to:

Preview: -$1000000000.00

The **Preview** field at the bottom of the tab allows a preview of how the variable of string data-type will be presented, allowing you to experiment with different formats.

The design of the pages of the Account Statement is now complete.

⊕ Return to the Workflow window to add an Output module.

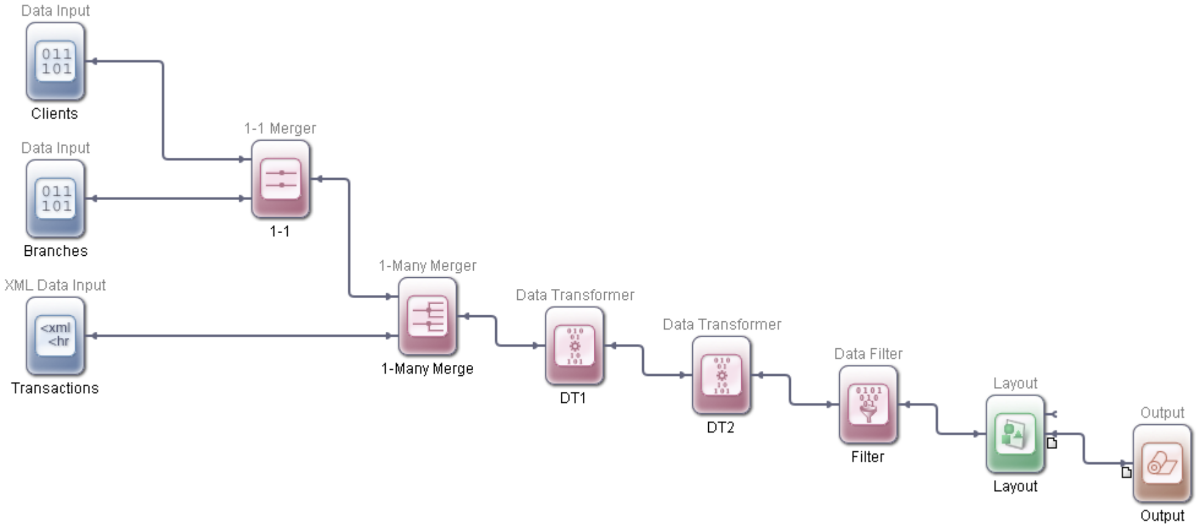
Click on the Workflow sub-tab of the workflow's tab .

The Workflow window opens.

5.2.2.10 Output

⊕ Define the media and the device on which the designed letter should be printed.

Select Output from the Module Tree, drag it to the Workflow Area, drop it and connect it to the Layout module.



⊕ Configure the Output Module.

Double-click on the new module.

Remember: The role of the Output module is to define printing media and device settings.

In this section we will define:

1. The media on which the workflow should be printed
2. Parameters for the printing device by which the workflow should be printed

Media

In our example, we only want to print the design of both pages to one kind of media (e.g. A4 size paper).

Remember: We assigned sheet name 1 to be the same (e.g. *A4*) for both of our designed pages.

⊕ Create a new Media Page to define the media that the letter should be printed on.

Click on the **Fill From Layout** button at the bottom of the dialog. A new media page using the name you entered as sheet name 1 for the pages in Layout (e.g. *A4*) will be created automatically.

Remember: When clicking on **Fill From Layout** a media page is created for each different sheet name 1 defined for a page in Layout. All media page attributes (**Sheet Name 1**, **Size**, **Width** and **Height**) will be filled according to the settings made in Layout.

Device

⊕ Define a profile for the printing device that the workflow should be printed on.

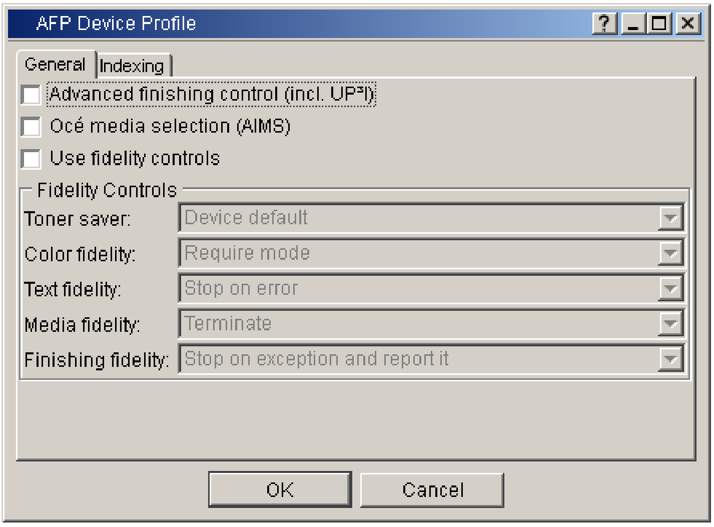
Click on the **New** button in the Device Profiles field of the dialog.

The Create Device Profile dialog opens.

⊕ Set the type of printer that will be used and give this profile a name.

Select the printer type (e.g. *AFP*) from the **Type** combo-box menu, enter a name (e.g. *AFP B*) in the **Name** edit-box and click on **OK**.

The AFP Device Config dialog opens:



AFP Device Profile

General | Indexing

□ Advanced finishing control (incl. UP³I)
□ Océ media selection (AIMS)
□ Use fidelity controls

Fidelity Controls

Toner saver:	Device default
Color fidelity:	Require mode
Text fidelity:	Stop on error
Media fidelity:	Terminate
Finishing fidelity:	Stop on exception and report it

The device config for the AFP engine allows the creation of an index using sheet names that have been pre-set in the Layout module.

Remember: Sheet names with index *3* and *4* were created in Layout and linked to variables that would return values from the records when they are printed.

⊕ Set up an index that will use the sheet names created in the Layout module to write an index to the AFP file that will be produced when the workflow is printed.

On the Indexing tab, activate the **Indexing** check-box and then click on the **Create New Child** icon twice to add two dependent rows to the Group Indexing category of the indexing area.

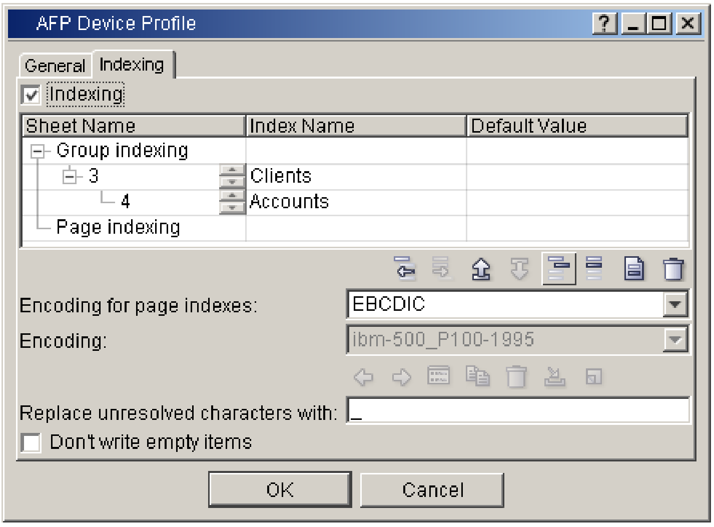
Two index definition rows are added.

The hierarchy of the index is shown in the Sheet Name Index column, it is best to create this according to the hierarchy of the records that are being printed. In our example, the record of the account is dependent on the record of the client and so we should reflect that in our index, i.e. the client's sheet name index (*3*) should be the first level of index and the account's sheet name index (*4*) should be on the second.

⊕ Set each of the indexes so that they directly relate to the sheet names that have been set in the Layout module.

Add the numbers of the created sheet name indexes (e.g. *3* and *4*) to the **Sheet Name Index** column and add corresponding names for each of these indexes that will be added to the spool file.

The indexes that will appear in the AFP spool file are defined:



☑ Indexing

Sheet Name	Index Name	Default Value
Group indexing		
3	Clients	
4	Accounts	
Page indexing		

Encoding for page indexes: EBCDIC

Encoding: ibm-500_P100-1995

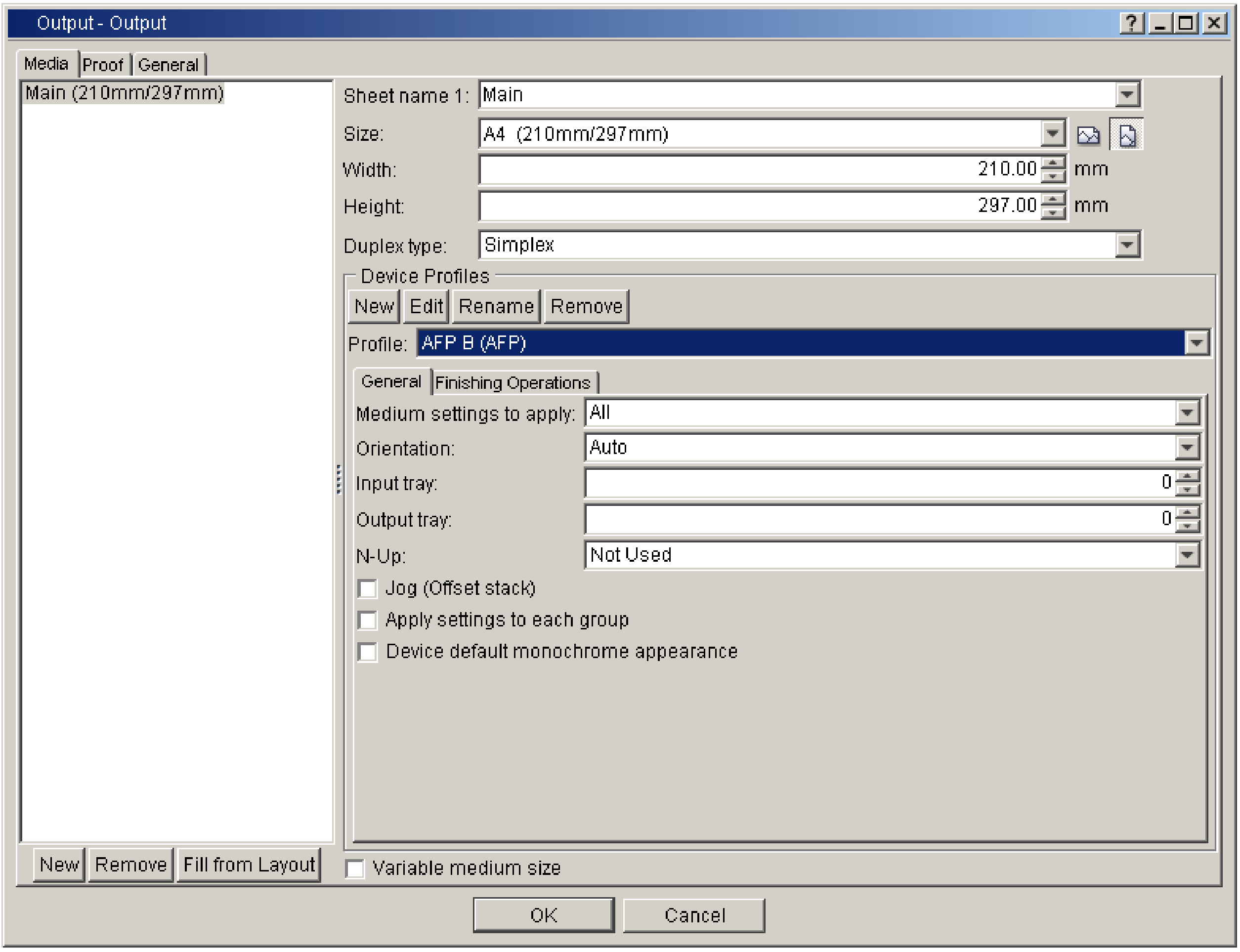
Replace unresolved characters with: _

□ Don't write empty items

⊕ Save the printer media configuration and dismiss the dialog.

Click on **OK**.

The new device profile is created and selected in the **Profile** combo-box:



Main (210mm/297mm)

Sheet name 1: Main

Size: A4 (210mm/297mm)

Width: 210.00 mm

Height: 297.00 mm

Duplex type: Simplex

Profile: AFP B (AFP)

Medium settings to apply: All

Orientation: Auto

Input tray: 0

Output tray: 0

N-Up: Not Used

□ Jog (Offset stack)

□ Apply settings to each group

□ Device default monochrome appearance

□ Variable medium size

⊕ Return to the Workflow window.

Click on **OK**.

The Output module closes.

5.2.2.11 Workflow Completion

The workflow is now complete.

⊕**Save** the workflow and check that the design would print how it is expected to.

Go to the menu **File | Save** and then click on the **Proof** icon to proof the workflow.

The Proof tab opens with the workflow processing.

5.2.3 Proof

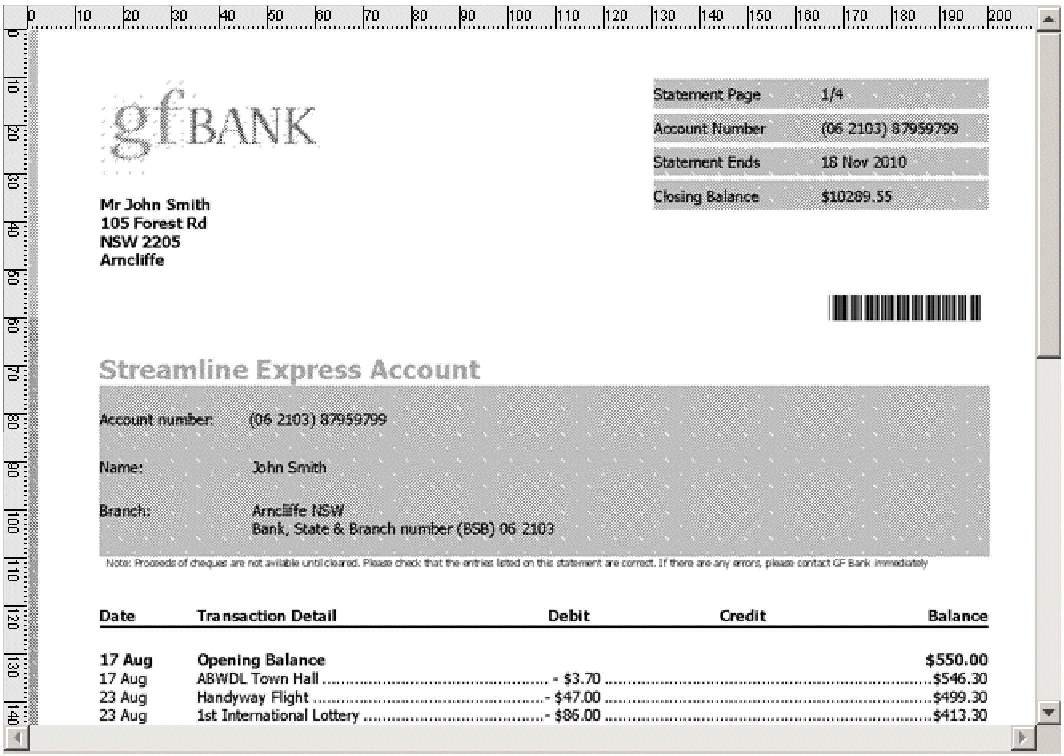
In our example, we are planning to print on an AFP printer by first producing a spool file written in the AFP page definition language to communicate the workflow's design. AFP belongs to the 'raster' class of page definition languages and, therefore, we would prefer to view the proofed workflow as a raster image.

⊕ Use the Create Profile dialog to create a new raster-type proof profile and view the design as a raster image.

In the Proof Profiles toolbar on the top-right-hand side of the Sheet Proof window select *Create New Profile* from the combo-box.

The Create Profile dialog opens. Enter a name for the profile in the Name edit-box (e.g. Raster1), select the *Raster* radio-button and click on **OK**.

A raster type of proof profile is created and automatically selected so that the view in the Sheet Proof Area is a raster image:



gf BANK

Statement Page	1/4
Account Number	(06 2103) 87959799
Statement Ends	18 Nov 2010
Closing Balance	$10289.55

Mr John Smith
105 Forest Rd
NSW 2205
Arncliffe

Streamline Express Account

Account number:	(06 2103) 87959799
Name:	John Smith
Branch:	Arncliffe NSW Bank, State & Branch number (BSB) 06 2103

Note: Proceeds of cheques are not avilable until cleared. Please check that the entries listed on this statement are correct. If there are any errors, please contact GF Bank immediately

Date	Transaction Detail	Debit	Credit	Balance
17 Aug	**Opening Balance**			**$550.00**
17 Aug	ABWDL Town Hall	- $3.70		$546.30
23 Aug	Handyway Flight	- $47.00		$499.30
23 Aug	1st International Lottery	- $86.00		$413.30

By default, a raster profile's settings produce a black and white view, but the profile can be adjusted. In our example, we intend to produce a color output and so would like to proof the design in color.

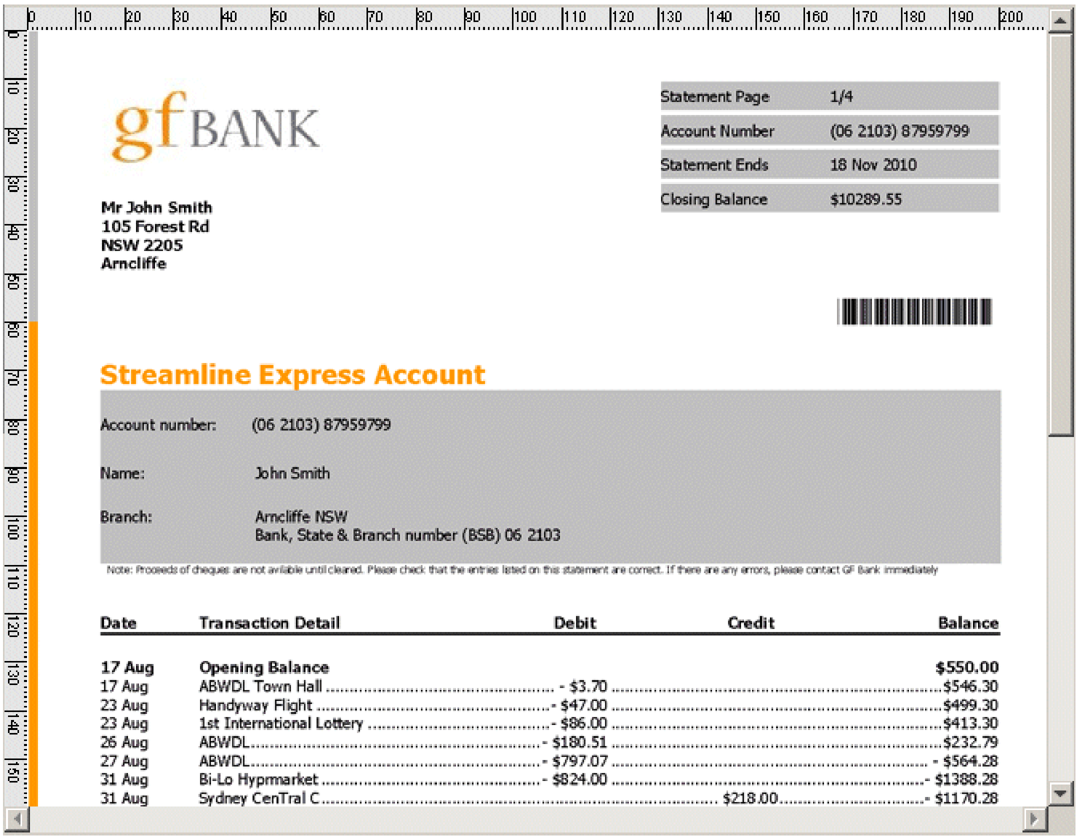
⊕ Adjust the properties of the raster profile to allow a color view of the design.

Click on the **Edit** icon in the Proof Profiles toolbar and, in the Edit Profile dialog that opens, select *CMYK* from the **Output colors** combo-box menu.

⊕ Confirm the adjusted Proof Profile configuration and view the new proof.

Click on **OK**.

The view in the Sheet Proof Area is now a raster image in CMYK color:



gf BANK

Statement Page	1/4
Account Number	(06 2103) 87959799
Statement Ends	18 Nov 2010
Closing Balance	$10289.55

Mr John Smith
105 Forest Rd
NSW 2205
Arncliffe

Streamline Express Account

Account number: (06 2103) 87959799

Name: John Smith

Branch: Arncliffe NSW
Bank, State & Branch number (BSB) 06 2103

Note: Proceeds of cheques are not avilable until cleared. Please check that the entries listed on this statement are correct. If there are any errors, please contact GF Bank immediately

Date	Transaction Detail	Debit	Credit	Balance
17 Aug	**Opening Balance**			**$550.00**
17 Aug	ABWDL Town Hall	- $3.70		$546.30
23 Aug	Handyway Flight	- $47.00		$499.30
23 Aug	1st International Lottery	- $86.00		$413.30
26 Aug	ABWDL	- $180.51		$232.79
27 Aug	ABWDL	- $797.07		- $564.28
31 Aug	Bi-Lo Hypmarket	- $824.00		- $1388.28
31 Aug	Sydney CenTral C		$218.00	- $1170.28

5.2.3.1 Data Proof

⊕ Check the Data-Proof window to ensure all records are being processed correctly.

Click on the Data sub-tab of the Proof environment Proof Sheet Data.

The Data Proof window opens.

Remember: You can choose different data processing stages in the workflow by selecting modules from the **Input** toolbar.

If an error is visible in either the Data Proof Area or the Log window-pane you should return to the Workflow window to solve the problem using all of the previous steps for guidance.

5.2.4 Production

⊕ Load the approved workflow to Production so that it can be converted to a suitable page description language for the printer that is to be used.

Return to the Workflow window using the tab ExB * Workflow Layout and click on the **Production** icon.

The Production environment opens with the workflow loaded.

Remember: Ensure that the correct workflow is loaded (in the **Workflow** combo-box) and that the pages to be printed (in the **Job Config** area) are set as required.

⌖ Select the type of page description language you would like the print job to use.

In the **Engine** combo-box select the desired output (e.g. *AFP*).

AFP is usually selected by default so no change will probably be necessary.

⌖ Create a printer engine profile to be configured to the output requirements.

Click on the **New** button in the Engine Config field of the Production window. Then, enter a name for the profile in the **Name** edit-box of the Create Engine Profile dialog that opens and click on **OK**.

The engine profile is created and can be selected in the **Profile** combo-box whenever the same engine type (e.g. AFP) is selected in the **Engine** combo-box.

⌖ Edit the AFP Engine Configuration Profile.

Click on the **Edit** button.

The AFP Engine Config dialog opens.

The AFP Engine Config dialog allows the editing of the file processing settings for the AFP engine. The attributes relate directly to the output format. You will recognize this if you are familiar with the format and can adjust the settings according to your requirements. In our example, we have color that we want to be visible in the print job and so we need to adjust the default settings to accommodate that.

⌖ Set the configuration so that the colors of the design will be visible in the AFP spool file.

In the **Output colors** combo-box on the General tab, select *Full color*.

Full color print is defined for the engine profile.

⌖ Confirm the AFP configuration.

Click on **OK**.

The settings are saved and the dialog is dismissed.

⌖ Create a printer driver profile to be configured to the output requirements.

Click on the **New** button in the Driver Config field of the Production window. Then, enter a name for the profile in the **Name** edit-box of the Create Driver Profile dialog that opens and click on **OK**.

The driver profile is created and can be selected in the **Profile** combo-box whenever the same engine type (e.g. *AFP*) is selected in the **Engine** combo-box.

Remember: There are substitution characters available that create a file name based on workflow parameters (e.g. %n is the name of file; %e is the default extension of the engine type selected).

⌖ Give the file to be produced a name.

Enter a path and a name (e.g. C:\Documents\%n.%e will create file ExB.afp in the Documents directory on the c: drive) in the **File name** edit-box.

The output file will be given a name when it is produced.

⌖ Produce the workflow.

Click on **Start**.

The Jobs window opens.

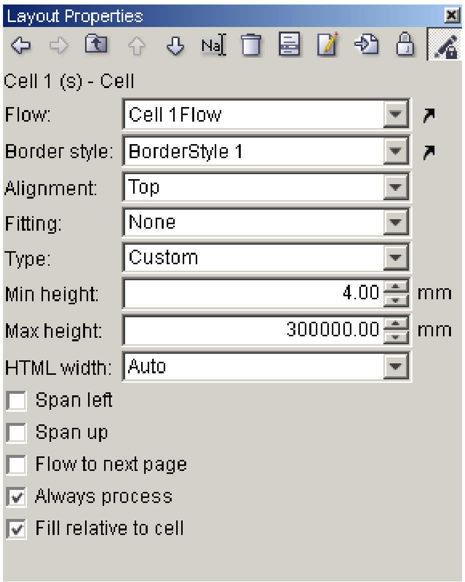
⊕ Collect the finished job from the location that was selected in the **File name** edit-box of the Production window.

When the job's log stops spooling and if no errors occur, go to the directory (e.g. C:\Documents) and open the file (e.g. ExB.afp).

Another software application (e.g. an AFP viewer) must be used to view the file.

Glossary

Cell Cells are the simplest units of a table. The properties of cells determine the content and appearance of the table that contains them:



Cell 1 (s) - Cell

Flow: Cell 1Flow

Border style: BorderStyle 1

Alignment: Top

Fitting: None

Type: Custom

Min height: 4.00 mm

Max height: 300000.00 mm

HTML width: Auto

□ Span left

□ Span up

□ Flow to next page

☑ Always process

☑ Fill relative to cell

The **Flow** combo-box assigns which flow will be shown in the cell and the **Border style** selects the border style for the cell. Some other commonly used attributes are **Type**, which determines whether the height of the cell should be fixed or according to content, and **Span left/ Span up** which indicate that the cell is merged with the one next to it.

Color Colors are content defining objects. They can be used as the content of fill *styles* which, in turn, are used by other objects.

Each color that is employed or created in Layout (except those that are components of images) is present as an object. These color objects can then be redefined as required and the color will be altered in all places where it is employed throughout the design.

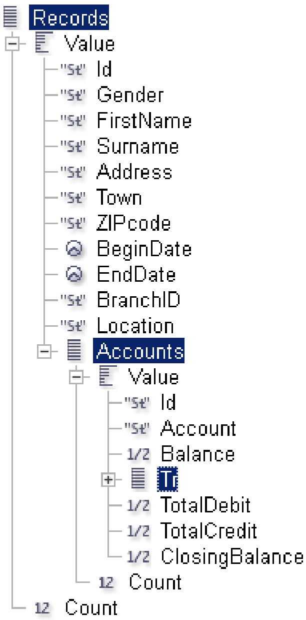
Data-Type The data-type of a variable defines the kind of data it can hold or how that data is to be stored.

Type becomes more important when a manipulation of the data is required. For example, if data is not of a numerical data-type (e.g. double) it cannot be used in calculations. In Inspire Designer, it is quite usual that data will be read into a workflow as String data-type and then converted to other data-types if it is required. Conversions can be made in either a Data Transformer or Layout module.

Some important data-types:

- String – Simple text characters. No formatting.
- Double – Numerical. A real number between - 5.0 x 10^{-324} and 1.7 x 10^{308} accurate to approximately 15 significant figures.
- Integer – Numerical. A whole number between -2147483648 and 2147483647.
- Boolean – A code for indicating whether a condition has been met. Field entries must be in the form of either: *True / False*; or *1 / 0*.

- Array – One data entry that governs a series of values that are dependent on that one entry and can only be accessed via that one entry.



```
Records
  Value
    Id
    Gender
    FirstName
    Surname
    Address
    Town
    ZIPcode
    BeginDate
    EndDate
    BranchID
    Location
    Accounts
      Value
        Id
        Account
        Balance
        Tr
        TotalDebit
        TotalCredit
        ClosingBalance
      Count
  Count
```

The screenshot above shows an example of *variables* of array data-type (high-lighted) as they would be seen in the Layout Tree. Notice how each has its own series of value returning variables that are dependent upon it.

Each array variable in Layout also contains a Count variable which calculates how many sets of values are returned by the variables. For record variables, this is the amount of records that is being read from the data file.

Device A device is the printing apparatus or software via which a workflow will be printed to a *media*.

In most cases, it is not a component of Inspire Designer and it is the purpose of the Production environment to communicate a print job with the separate device using a chosen page definition language.

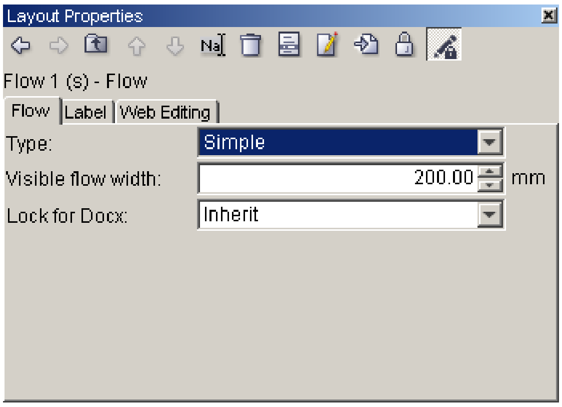
Extra commands for a printing device that are not part of a document design (e.g. finishing operations) can also be defined in a workflow via the Output module.

Flow Flows are text passages.

Flows also become the containing objects for other objects nested within them, e.g. an image can be directly placed at the cursor point in the flow. This includes other flows, e.g. many flows can be nested inside one flow, allowing one passage of text to be built from many passages of text that can each be manipulated (i.e. removed or replaced) independently and as required.

Like other content defining objects, one advantage of having flows as separate objects to the area in which they are used is that the content can be swapped easily. This becomes apparent when using more advanced types of flow.

Flows also have types which can be chosen from any selected flow's properties panel in the **Type** combo-box:

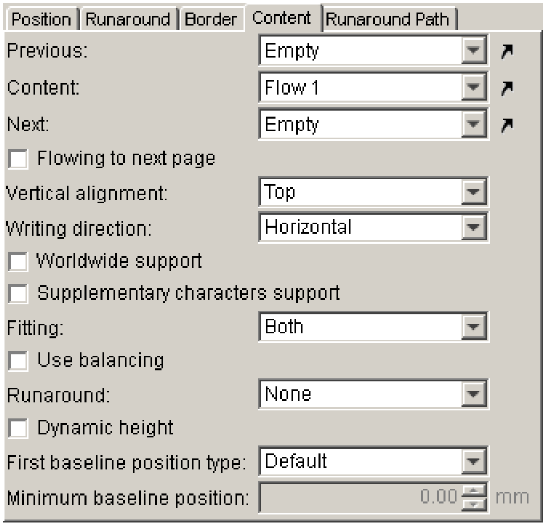


Type: Simple
Visible flow width: 200.00 mm
Lock for Docx: Inherit

The **Type** *Simple* is a single, static passage of text.

Another commonly used type is *Select by condition* which can be assigned variables (of Boolean *data-type*) to select alternative flows, if their conditions are met.

Flow Area Flow areas define positions of *flows*. Their Content tab relates to flow assignation:



Previous: Empty
Content: Flow 1
Next: Empty
□ Flowing to next page
Vertical alignment: Top
Writing direction: Horizontal
□ Worldwide support
□ Supplementary characters support
Fitting: Both
□ Use balancing
Runaround: None
□ Dynamic height
First baseline position type: Default
Minimum baseline position: 0.00 mm

The **Content** combo-box is key on this tab: The drop-down menu there reveals a list of all created flows, allowing a quick swap of the flow contained within the flow area. Other attributes of the Content tab relate to the alignment of the selected flow in relation to the flow area and to other flow areas. It is also possible to set an overflow coming from a **Previous** flow area and going on to a **Next** flow area.

Image Images are external files that are used as content defining objects. Most types of image file are supported by Inspire Designer with full-color (CMYK and RGB) and can be further manipulated to adjust their appearance and quality.

Once the images are inserted into Layout they appear, categorized, under the Images family, in the Layout Tree. Like all content defining objects they can be used throughout the design inside of image areas, flows or fill styles.

Image Area Image areas define positions for *images* when they are placed in Layout (e.g. on a page). Their Content tab relates to which image it contains and other settings:

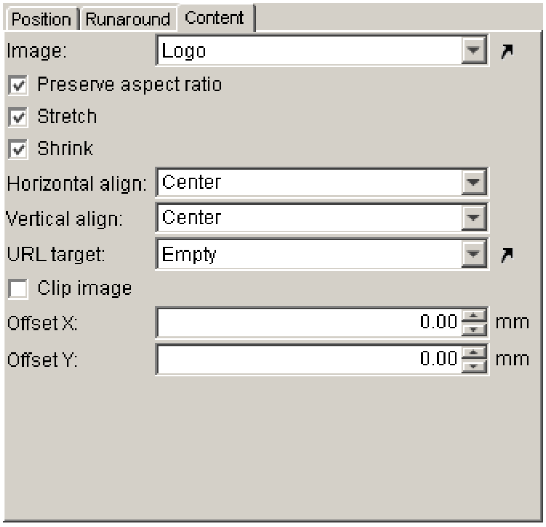


Image: Logo
☑ Preserve aspect ratio
☑ Stretch
☑ Shrink
Horizontal align: Center
Vertical align: Center
URL target: Empty
□ Clip image
Offset X: 0.00 mm
Offset Y: 0.00 mm

The **Image** combo-box menu reveals a list of all images inserted to Layout, allowing a quick swap of the image contained. Other attributes here relate to the alignment of the selected image in relation to the image area and what should happen to the appearance of the image when the size of the image area is altered, e.g. whether it should stretch with the area.

Layout Tree The Layout Tree is a list of the objects that are present in the Layout module and ready to be (or already have been) used in the design of the pages. Each family on the first level of the tree is an object category.

As each new object is inserted into the Layout or drawn in the Layout Area, corresponding nodes are added to the Layout Tree according to their family. By expanding the branches, the objects of each given family can be explored and their properties can be adjusted.

Media Media is the definition of the physical form on which a workflow will be printed, e.g. A3 paper in landscape orientation.

It is not an actual property of a workflow because media selection is, in most cases, controlled by the *device* on which the workflow is printed, but certain parameters in the workflow can be adjusted in preparation for the intended printing media. E.g. collation of *sheets* to be printed on the media with impositioning modules and media size in the Output module.

The media is represented by *pages* in the Layout module for the purpose of design.

Module Modules are the components of a workflow, each module has a particular function and can be selected if that function is required. These functions are categorized according to the requirements of a workflow:

- Data Input Modules – They are, most often, the very first modules of a workflow. Their role is to read differing types of data file into the workflow.

  The configuration of data input usually involves entering the location of the data file to be read and then confirming some parameters about its format. As the data is read into the workflow, each field (or element) is declared as a data variable. These variables can then be used in the workflow for manipulations and printing as a part of the designed document.

  Some of the data input modules allow a preview of the structure of the data and of the variables' *data-type*.

- Data Processing Modules – They are designed to perform differing manipulations with any data that has been read into the workflow. Which data processing modules to use and which data variables to manipulate with them, depends on the intended use for the data.

One type of data processing modules is those that combine two data inputs, i.e. 1-1 Merge, 1-Many Merge, Data Concatenator and Data Add. These have different methods of joining the data to be related within the common set of records or in an unrelated manner. Other modules (e.g. Data UnGroup) can then be added to manage the relationship between data.

The Data Transformer module allows selected data variables to have their *data-type* converted and also create new variables from the existing ones (e.g. in yet another format or selecting data from them according to a condition).

Filters (i.e. Data Filter and Record Filter) allow the removal of data that is not useful for the workflow design and others (e.g. Text Replacer and Data Precise) help with automated correction of data that might not be presented as required.

- Layout Modules – They play an important part in most workflows by defining the design of the intended printed output. Pre-defined layouts can be added to the workflow as XSLFO or TNO (Inspire Designer's own output format) files and the Layout module has its own page editing environment.
- Impositioning Modules – They provide a variety of ways to collate designed pages; i.e. arranging how they should be printed on a medium, how they should be arranged on sheets of paper.

  There are modules that combine different designs (e.g. Sheet Concatenator and Sheet Duplexer) so that they will be printed in some sequences (e.g. one sheet from one design and one sheet from the next) and modules that can filter, sort or select certain sheets from a single Layout according to given criteria.

  Using impositioning modules allows for greater flexibility when planning how to print a workflow. For example, there might be two types of printing medium and a different sheet arrangement required for each. To solve this, after a Layout module, you could enter two different impositioning module set-ups to two different outputs. Then later, in the production stages, the output can be selected according to the preferred medium.

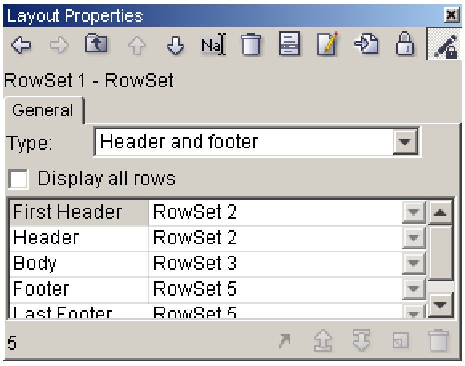
  Some of the impositioning modules use sheet names, that have been assigned to the pages in Layout, to select which sheets to collate.
- Output Modules – They define how, at the close of the workflow, the details of the workflow's design will be passed-out.

  The Output module can be used for closing any workflow with a design to be printed. In it, specific considerations about the intended media and printing device to be used can be defined.

  Alternatively there is: the Data Output module, to create data files with the records that have been read into and manipulated by the workflow; and other modules that provide statistics on the processed sheets or print jobs.

Page Pages are the main objects in the Layout module that define the design space. They represent the media on which the workflow will be printed. The properties of pages relate to the size of the design space, the order in which the pages will appear when the workflow is processed and the subsequent repetition of that page order if it is required.Each page is a space on which the other objects of the design are positioned. As new objects are drawn or added to the page, corresponding nodes are added to the branch of that page in the Layout Tree. Order is important on this Pages branch: Objects that are closer to a page's node in the Layout Tree will be processed before those that are further away. The object order can be optimized.

Row set Row sets are the building blocks of tables. They group together cells to make a table. A row set's type and the other dependent row sets that it contains can be managed in its properties panel:



Layout Properties

RowSet 1 - RowSet

General

Type: Header and footer

☐ Display all rows

First Header	RowSet 2
Header	RowSet 2
Body	RowSet 3
Footer	RowSet 5
Last Footer	RowSet 5

5

Commonly used types of row set include:

- *Single row* – A collection of cells that span across the table, one cell per column.
- *Multiple rows* – A grouping of *single row* type row sets, defining that they will always be shown together in the design.
- *Repeated* – A row set that will be repeated for each value of an array that is read from a data file.
- *Header/Footer* – A row set that contains the row sets of a whole table with separately allocated header and footer row sets. It further assumes that one table might have to overflow from one defined position (flow area) to another and that, in such cases, it might be necessary to have differing headers and footers to show the continuation. Therefore, it is made up of a First Header row set (only in the first flow area), a Header row set (in all following flow areas), a Body row set, a Footer (in all flow areas except the last) and a Last Footer (only in the last flow area, i.e. at the absolute end of the table).
- *Select by condition* – A row set that governs a collection of other row sets to determine which will be shown if the condition in an assigned variable (of Boolean data-type) is met.

The **Display all rows** check-box on the properties panel determines that all of the rows in the row set will be displayed in the design in the Layout Area. If it is not checked, only one variant of any conditionally selected rows will appear. For example, First Header or Header is selected by a condition that determines if it is an overflow header or not, therefore only one of the headers (First Header) will appear in the Layout Area unless **Display all rows** is checked.

Sheet Sheets are the basic unit of pages that have been processed. When a page is processed it creates a sheet and if the same page is processed again it creates another sheet, i.e. one sheet is created from every single page repetition.

For example, if we have 3 pages in Layout that will be repeated for 3 data records that have been read into Layout (3 pages repeated 3 times) then we will create 9 sheets.

Sheets are then collated by impositioning modules in preparation for a printing media. However, collation does not change the definition of the sheet, e.g. 2 sheets that are collated as duplex are still 2 individual sheets.

Style Styles are content defining objects in the Layout module. They include:

- Paragraph Styles
- Text Styles
- Fill Styles
- Border Styles
- Line Styles

Whenever a new style is used by an object, a style is created as an object. Different types of style have different attributes that can be adjusted in their individual properties panels, but once created they exist to be selected for use by other objects throughout the design:

- Selected text of a flow can adopt existing text styles or paragraph styles using the Text Style and Para Style toolbars at the top of the Layout window.
- Shapes can adopt line styles and fill styles for both its outline and fill.
- Cells can adopt border styles which employ both line styles and fill styles.

Table Tables are content defining objects in the Layout module. A table is an arrangement of flows, each flow inside an individual cell. Cells are then organized into *row sets* and these row sets are then organized into another larger row set which collects together all of the row sets of the table.

A table defines which row set it contains and parameters relating to the style of the whole table:

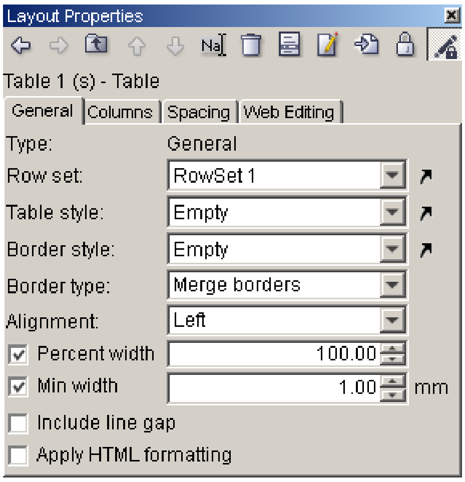


Table 1 (s) - Table

Type: General
Row set: RowSet 1
Table style: Empty
Border style: Empty
Border type: Merge borders
Alignment: Left
☑ Percent width 100.00
☑ Min width 1.00 mm
☐ Include line gap
☐ Apply HTML formatting

Using this properties panel, border styles and alignment to the position defining object (flow area) can be applied to the whole table. Columns are also a style defining parameter which can be set on the Columns tab.

Variable Different kinds of variable-data in a workflow are represented by variables. They are data objects that are read repeatedly when a workflow is processed and, as the workflow continues to process, can change according to some criteria.

All variables are present as objects in the Layout module for use as part of text passages (flows) in a document's design. Each variable can be placed in the design where it will return values of a certain *data-type*. The format of those values is adopted from the text style applied to them.

Variables can also be used by other objects (e.g. flows, images) to determine a selection between different objects, creating variable objects.

There are three categories of variables:

1. Record Variables – Are read from the different fields of a data file that is read to a workflow by a data input module. Each field of the file is defined as a variable, i.e. an object that returns the value of that field for each record being read. By default, these variables are named according to the titles of the data fields that they correspond to.

   They then pass along the workflow and can be manipulated and can have new record related variables calculated from them. Finally, they are present as objects in the Layout module and can be found categorized under Data/Records in the Layout Tree.

   From here they can then be used as part of a text passage (flow) in the design. The content that they return is the value of the corresponding field or element in the data file. Considering that each designed page is, by default, repeated for each record that is read, then the variable's value will actually be different for each printed sheet.

2. System Variables – Calculations that will be performed by Inspire Designer when a workflow is processed. They return values that relate to this workflow. They can be found categorized under Data/SystemVariable in the Layout Tree.

   This data can also be placed in the design to print information that relates to the actual print job. For example, PageCounter shows how many pages have been printed and returns the page number for each. This is useful for adding individual page numbers to the pages printed.

3. User Defined Variables – Can be created in chosen positions on the data branch of the Layout Tree. The most frequently used kinds are:

   - *Calculated* – A script defined variable, usually a brief calculation expression. The value can then be used to be printed or be assigned elsewhere.

   - *Global* – A variable that is linked to another variable and used to remember the value of that variable until the next time it is processed, whereupon the new value of that variable will be added to previous value, creating an accumulative sum.

Index