# GREATIS PRINT SUITE
# and
# GREATIS PRINT SUITE PRO

## User Manual

**Copyright © 2001 Greatis Software**

**Contents** 1
**About Greatis Print Suite** 4
**New in Greatis Print Suite** 6
**Print Suite and Print Suite Pro hierarchy** 9
**TPrintJob** 10
Hierarchy 10
Interface 10
Methods 10
Properties 13
Events 14
Other declarations in PSJob.pas 16
Sequence of events in TPrintJob 17
How to realize print job with TPrintJob 18
How to create custom print job class 19
**Print Jobs** 22
TCustomGridPrintJob 24
Hierarchy 24
Interface 24
Methods 24
Properties 25
TSimpleGridPrintJob 26
Hierarchy 26
Interface 26
Methods 26
Properties 27
Events 28
TCustomTextGridPrintJob 29
Hierarchy 29
Interface 29
Methods 29
Events 30
TSimpleTextGridPrintJob 31
Hierarchy 31
Interface 31
Methods 31
Properties 32
Events 32
TSpecialTextGridPrintJob 34
Hierarchy 34
Interface 34
Methods 34

TStringGridPrintJob 35
Hierarchy 35
Interface 35
Methods 35
Properties 36
Events 36
TDBGridPrintJob 37
Hierarchy 37
Interface 37
Methods 37
Properties 38
Events 39
TListViewPrintJob 40
Hierarchy 40
Interface 40
Methods 40
Properties 42
Events 43
TStringsPrintJob 44
Hierarchy 44
Interface 44
Methods 44
Properties 45
Events 45
TMultiPrintJob 46
Hierarchy 46
Interface 46
Methods 46
Properties 47
Events 48
TDraftPrintJob 49
Hierarchy 49
Interface 49
Methods 49
Properties 49
Events 50
**TPreview** 51
Hierarchy 51
Interface 51
Methods 51
Properties 52
Events 53

**TPreviewToolbar** 54
Hierarchy 54
Interface 54
Methods 54
Properties 55
Events 55
**TPreviewStatusBar** 56
Hierarchy 56
Interface 56
Methods 56
Properties 56
Events 57
**TPreviewComboBox** 58
Hierarchy 58
Interface 58
Methods 58
Properties 58
Events 59
**TPreviewLabel** 60
Hierarchy 60
Interface 60
Methods 60
Properties 60
Events 61
**TPreviewWindow** 62
Hierarchy 62
Interface 62
Methods 62
Properties 62
Events 64
**How it works?** 65
**Customization and localization** 66
**Demo and examples** 68
**Contact Information** 73

# About Greatis Print Suite

**Greatis Print Suite** is a set of components which provides advanced print and preview features into Delphi and C++ Builder applications. The suite contains:

- **TPrintJob**
  main non-visual component of the suite which provides easy multipage printing
- **TPreview**
  control which provides easy preview
- **TPreviewToolbar**
  ready-to-use toolbar which provides control of preview
- **TPreviewStatusBar**
  ready-to-use status bar which provides display preview parameters
- **TPreviewComboBox**
  combo box which provides display and control of preview scale
- **TPreviewLabel**
  label to display preview parameters
- **TPreviewWindow**
  ready-to-use preview window

**Greatis Print Suite Pro includes additional TPrintJob descendants:**

- **TSimpleGridPrintJob**
  component for easy graphics grid printing by responding to events
- **TSimpleTextGridPrintJob**
  component for easy text grid printing by responding to events
- **TStringGridPrintJob**
  component for easy TStringGrid printing
- **TDBGridPrintJob**
  component for easy TDBGrid printing
- **TListViewGridPrintJob**
  component for easy TListView printing
- **TStringsPrintJob**
  component for easy text and TStrings contents printing
- **TMultiPrintJob**
  component for integrate other print jobs
- **TDraftPrintJob**
  component for draft printing

All components are compatible with Delphi 3,4,5 and C++ Builder 3,4,5.

To install suite add PrintSuite.pas (and PrintJobs.pas for Pro version) into any package and recompile it. You can install any component of the suite individually by adding its source file into any package and recompiling it.

Component modules:

| | |
|---|---|
| TPrintJob | PSJob.pas |
| TPreview | PSPreview.pas |
| TPreviewToolbar | PSToolbar.pas |
| TPreviewStatusBar | PSStatusBar.pas |
| TPreviewComboBox | PSCombo.pas |
| TPreviewLabel | PSLabel.pas |
| TPreviewWindow | PSWindow.pas |

! Components and modules listed below are available only in Print Suite Pro !

| | |
|---|---|
| TSimpleGridPrintJob | PJSimpleGrid.pas |
| TSimpleTextGridPrintJob | PJSimpleTextGrid.pas |
| TStringGridPrintJob | PJStringGrid.pas |
| TDBGridPrintJob | PJDBGrid.pas |
| TListViewPrintJob | PJListView.pas |
| TStringsPrintJob | PJStrings.pas |
| TMultiPrintJob | PJMulti.pas |
| TDraftPrintJob | PJDraft.pas |

**Greatis Software thanks Ralf Ebert**
**(www.geotrainer.com)**
**whose perfect ideas are used in this suite**

# New in Greatis Print Suite

# Version 1.31

New buttons in TPreviewToolbar: btnWholePage and btnPageWidth.
New property in TStringsPrintJob (Print Suite Pro): WordWrap.

# Version 1.3

New component in Print Job package (Print Suite Pro): TDraftPrintJob.
TDraftPrintJob is a ready-to-use component for draft printing. It allows to place up to 100 pages from another print job on one physical page. Just link TDraftPrintJob with another print job and set scale factor. Scale factor may be from 2 (4 pages on one) to 10 (100 pages on one).

## New in TPrintJob

### Methods

```
procedure LockJobsUpdate;
procedure UnlockJobsUpdate;
```
These methods disable and enable linked print jobs updating (see also AddPrintJobNotification and DeletePrintJobNotification).

```
procedure LockControlsUpdate;
procedure UnlockControlsUpdate;
```
These methods disable and enable linked controls updating (see also AddControlNotification and DeleteControlNotification).

### Properties

```
property UpdateJobsLocked: Boolean;
```
This read-only public property contains linked print jobs update lock state.

```
property UpdateControlsLocked: Boolean;
```
This read-only public property contains linked controls update lock state.

# Version 1.2

New component in Print Job package (Print Suite Pro): TMultiPrintJob. TMultiPrintJob is a ready-to-use component for integrating other print jobs into one multi-job. Just add any print jobs into PrintJobs list of TMultiPrintJob and TMultiPrintJob will print they all in specified order.

# Version 1.1

## New Print Jobs package

Print Jobs package contains new ready-to-use components for print graphic and text grids, grid components (like TListView and TStringGrid), databases and text files. This package is a part of Greatis Print Suite Pro.

## New in TPrintJob

**Methods**

```
procedure LockUpdate;
procedure UnlockUpdate;
```
These methods disable and enable print job updating.

```
procedure LockDraw;
procedure UnlockDraw;
```
These methods disable and enable print job drawing.
Lock state is determines by internal lock counters. LockXXX procedure increments, and UnlockXXX procedure decrements these counters if they value is greater than zero.

```
procedure Abort; virtual;
```
This method terminates the printing of a print job.

```
function DPIX: Integer;
function DPIY: Integer;
```
These functions return horizontal and vertical resolution of current printer in pixels per inch.

```
procedure DrawBitmap(TheCanvas: TCanvas;
     const Rect: TRect; Bitmap: TBitmap;
     Target: TDrawTarget);
```
This function draws bitmap in stretch mode. For Target = dtPrinter Bitmap are converted to DIB (Device Independed Bitmap) before drawing. Use this function instead of StretchDraw.

```
procedure AddPrintJobNotification(
  APrintJob: TCustomPrintJob);
procedure DeletePrintJobNotification(
  APrintJob: TCustomPrintJob);
```
These are internal methods to link and unlink TPrintJob with another TPrintJob.

**Properties**

```
property UpdateLocked: Boolean;
```
This read-only public property contains update lock state.

```
property DrawLocked: Boolean;
```
This read-only public property contains draw lock state.

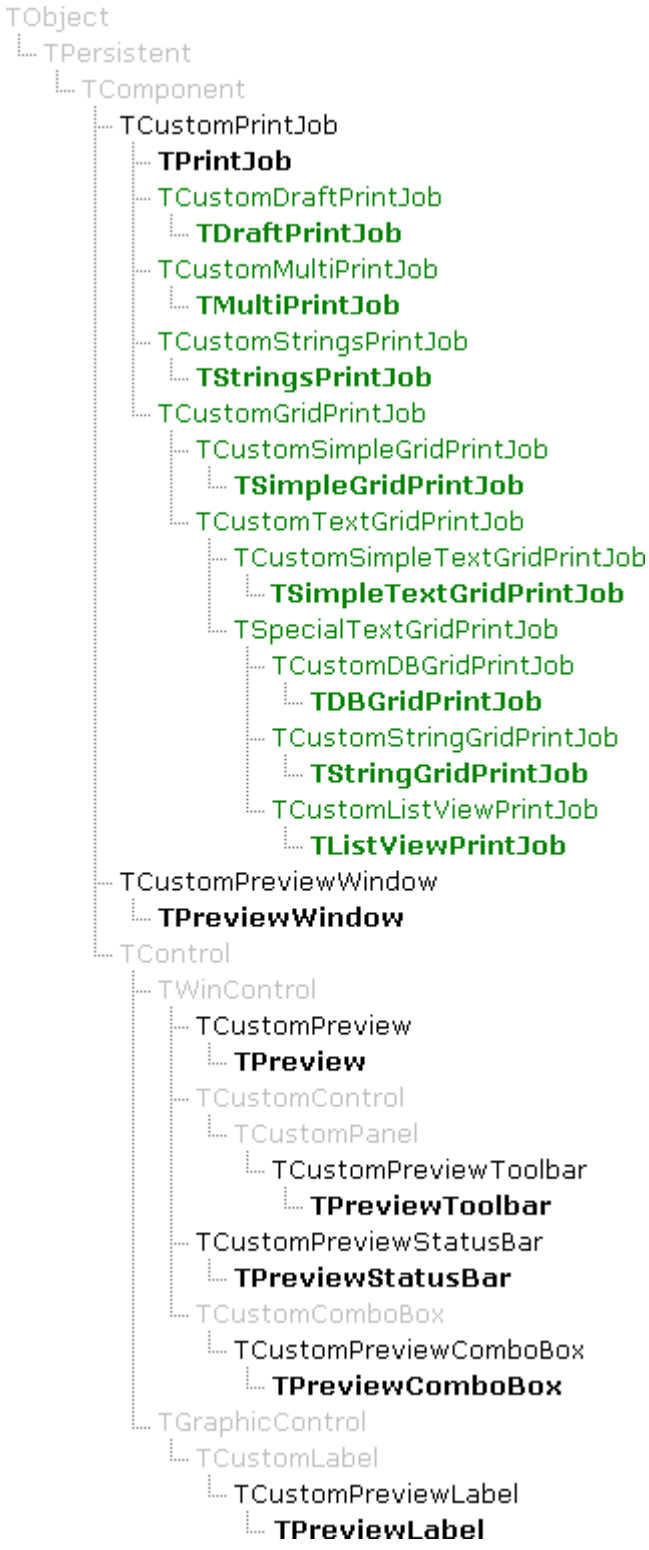
```
property Aborted: Boolean;
```
This read-only public property determines if the user aborted the print job.

## New in TPreview

**Methods**

```
function DPIX: Integer;
function DPIY: Integer;
```
These functions return horizontal and vertical resolution of preview current view scale adjusted. Don’t miss this functions with similar functions of TPrintJob

# Print Suite and Print Suite Pro hierarchy




**Legend:** Gray - Standard VCL classes; Green - Print Suite Pro classes; **Bold** - Published classes

# TPrintJob

TPrintJob (PSJob.pas) is a non-visual component for easy multipage printing. Using TPrintJob, you need not call BeginDoc, NewPage, EndDoc or other low-level print functions; just draw your pages content - TPrintJob will handle the rest. Now you need only to define OnDraw event handler and draw your page content within this handler using event parameters. TPrintJob has many useful methods, properties and events which can be used for component and printer setup, simplification of drawing, and notification of other components upon changes.

## Hierarchy

TObject
    TPersistent
        TComponent
            TCustomPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal objects. These methods call OnCreate and OnDestroy events too.

```
procedure LockUpdate;
procedure UnlockUpdate;
```
These methods disable and enable print job updating.

```
procedure LockDraw;
procedure UnlockDraw;
```
These methods disable and enable print job drawing.

```
procedure LockJobsUpdate;
procedure UnlockJobsUpdate;
```
These methods disable and enable linked print jobs updating (see also AddPrintJobNotification and DeletePrintJobNotification).

```
procedure LockControlsUpdate;
procedure UnlockControlsUpdate;
```
These methods disable and enable linked controls updating (see also AddControlNotification and DeleteControlNotification).

Lock state is determines by internal lock counters. LockXXX procedure increments, and UnlockXXX procedure decrements these counters if they value is greater than zero.

```
procedure Draw(TheCanvas: TCanvas; PageIndex: Integer;
  Target: TDrawTarget); virtual;
```
This method calls InitDraw and DrawAreas for header, main area and footer of page. Calling DrawArea for header and footer is effected from Options property.

```
function DPIX: Integer;
function DPIY: Integer;
```
These functions return horizontal and vertical resolution of current printer in pixels per inch.

```
procedure DrawBitmap(TheCanvas: TCanvas;
     const Rect: TRect; Bitmap: TBitmap;
     Target: TDrawTarget);
```
This function draws bitmap in stretch mode. For Target = dtPrinter Bitmap are converted to DIB (Device Independed Bitmap) before drawing. Use this function instead of StretchDraw.

```
procedure InitDraw(TheCanvas: TCanvas;
  Target: TDrawTarget); virtual;
procedure DrawArea(TheCanvas: TCanvas;
  PageIndex: Integer; Rect: TRect;
  Area: TDrawArea; Target: TDrawTarget); virtual;
procedure PrinterSetupChange; virtual;
procedure StartPrint; virtual;
procedure EndPrint; virtual;
procedure PrintProgress(CurPage,
  MinPage,MaxPage: Integer); virtual;
procedure StartPrintPage(PageIndex: Integer); virtual;
procedure EndPrintPage(PageIndex: Integer); virtual;
```
These methods just call corresponding event, but may be overridden in derived component. You need not to call them directly.

```
procedure Print; virtual;
procedure PrintDialog; virtual;
procedure PrinterSetupDialog; virtual;
```
These methods are main methods to work with TPrintJob. You can call them to print your job, call print dialog, or call the printer setup dialog. When user close print dialog or printer setup dialog by OK button, the TPrintJob calls OnPrinterSetupChange event.

```
procedure Abort; virtual;
```
This method terminates the printing of a print job.

```
function GetSheetRect: TRect; virtual;
function GetMarginRect: TRect; virtual;
function GetHeaderRect: TRect; virtual;
function GetPageRect: TRect; virtual;
function GetFooterRect: TRect; virtual;
```
These methods return rectangle for print sheet, sheet without margins, header, footer and main area of page.

```
procedure ResetToDefaultPage; virtual;
```
This method sets PageWidth and PageHeight properties to width and height of the default printer page.

```
function PrinterOK: Boolean; virtual;
```
This method checks availability of default printer and returns True if printer available otherwise False.

```
procedure Update; virtual;
```
This method update print job and all linked controls. You need not to call Update directly, but can override it to add your reaction on update in derived component.

```
procedure AddControlNotification(AControl: TControl);
procedure DeleteControlNotification(AControl: TControl);
```
These are internal methods to link and unlink controls with TPrintJob. All controls in Print Suite call these methods when their PrintJob property is changed.

```
procedure AddPrintJobNotification(
  APrintJob: TCustomPrintJob);
procedure DeletePrintJobNotification(
  APrintJob: TCustomPrintJob);
```
These are internal methods to link and unlink TPrintJob with another TPrintJob.

```
function InchToMm(Inches: Double): Double;
function MmToInch(Millimeters: Double): Double;
function InchToPix(Inches: Double;
  Direction: TDirection): Double;
function MmToPix(Millimeters: Double;
  Direction: TDirection): Double;
function PixToMm(Pixels: Double;
  Direction: TDirection): Double;
function PixToInch(Pixels: Double;
  Direction: TDirection): Double;
```
These are methods for converting units between inches - millimeters - pixels. Direction parameter using in pixels conversion methods are dirHorizontal or dirVertical. This additional parameter is needed because vertical and horizontal resolution on printer may be different.

```
function ConvertUnits(Source: Double;
  FromUnits,ToUnits: TUnits; Dir: TDirection;
  FullRange: Double): Double;
```
This method converts units between FromUnits and ToUnits. TUnits is an enumerated type and can be unInches, unMillimeters, unPixels or unPercents. In FullRange parameter 100% value for unPercents is passed.

```
procedure ConvertMargins(M: TMargins;
  FromUnits,ToUnits: TUnits);
```
This method converts margins between FromUnits and ToUnits.

**Properties**

**public**

```
property UpdateLocked: Boolean;
```
This read-only public property contains update lock state.

```
property UpdateJobsLocked: Boolean;
```
This read-only public property contains linked print jobs update lock state.

```
property UpdateControlsLocked: Boolean;
```
This read-only public property contains linked controls update lock state.

```
property DrawLocked: Boolean;
```
This read-only public property contains draw lock state.

```
property Aborted: Boolean;
```
This read-only public property determines if the user aborted the print job.

**published**

```
property PageCount: Integer;
```
Count of pages in print job.

```
property Title: string;
```
Print job title. This text is displayed in the Print Manager and on network header pages.

```
property Margins: TMargins;
```
Left, right, top and bottom page margins. These values are used in GetMarginRect, GetHeaderRect, GetFooterRect and GetPageRect methods and in Rect parameter of OnDraw event.

```
property MarginsUnits: TUnits;
```
Units for Margins property.

```
property Header: Double;
property Footer: Double;
```
Height of header and footer areas.

```
property HeaderUnits: TUnits;
property FooterUnits: TUnits;
```
Units for Header and Footer properties.

```
property PageMode: TPageMode;
```
Mode for PageWidth and PageHeight properties. May be pmDefault (TPrintJob page size is obtaining from printer page size) or pmCustom, when you can set the arbitrary page size.

```
property PageWidth: Double;
property PageHeight: Double;
```
Width and height of TPrintJob page. When PageMode is pmDefault they contain printer page size and available only for read otherwise you can set any values.

```
property PageUnits: TUnits;
```
Units for PageWidth and PageHeight properties. When PageUnits is unPercents page size is calculated in percents of printer page size.

```
property Orientation: TPageOrientation;
```
Page orientation. Can be orDefault when orientation is corresponds to current printer page orientation, or orLandscape/orPortrait for forced page orientation.

```
property Options: TJobOptionsSet;
```
Print job options. This set can contain joMarging to enable margins, joHeader to draw page header and joFooter to draw page footer.

All these properties are defined in public section of TCustomPrintJob class. TPrintJob just make they published.

**Events**

```
property OnCreate: TNotifyEvent;
property OnDestroy: TNotifyEvent;
```
These events occur when component are created and destroyed.

```
property OnInitDraw: TInitDrawEvent;
TInitDrawEvent = procedure (Sender: TObject;
  TheCanvas: TCanvas; Target: TDrawTarget) of object;
```
This event occurs when component are preparing to draw. Use this event to initialize TheCanvas parameters, common for all pages.

```
property OnDraw: TDrawEvent;
TDrawEvent = procedure (Sender: TObject;
  TheCanvas: TCanvas; PageIndex: Integer;
  Rect: TRect; Area: TDrawArea;
  Target: TDrawTarget) of object;
```
Main event of TPrintJob. Use this event to draw page contents. You must draw requested page only (PageIndex parameter)! In Rect are passed rectangle for area and in Area are passed area type that you must draw (header, footer or main area of page). Use Target value to determine draw target between printer and preview if you want to create different output (e.g. black and white printing and color preview). Page numbering is started from 1, thus first page index is 1 and last page index is PageCount.

```
property OnPrinterSetupChange: TNotifyEvent;
```
This event occurs when printer setup is changed. TPrintJob does not monitor real printer setup changing and call this event only after call PrinterDialog and PrinterSetupDialog own methods.

```
property OnStartPrint: TNotifyEvent;
property OnEndPrint: TNotifyEvent;
```
These events occur before start and after end of printing. You can use this events to display information about printing process.

```
property OnPrintProgress: TPrintProgressEvent;
TPrintProgressEvent = procedure (Sender: TObject;
  CurPage,MinPage,MaxPage: Integer) of object;
```
This event occurs after the printing of each page. Use this event to display print progress. MinPage and MaxPage are contained start and end page of current printing progress. This values can be different from 1 and PageCount when user print your document by PrintDialog method.

```
property OnStartPrintPage: TPageNotifyEvent;
property OnEndPrintPage: TPageNotifyEvent;
TPageNotifyEvent = procedure (Sender: TObject;
  PageIndex: Integer) of object;
```
Events for simple notification of start and end print page.

```
property OnUpdate: TNotifyEvent;
```
This event occurs after TPrintJob update.

**Note.**
All TPrintJob methods, properties and events are defined in TCustomPrintJob. TPrintJob just publish properties and events. Use TCustomPrintJob to create descendant components with hidden properties and/or events.

### Other declarations of PSJob.pas

### TMargins

Internal class for TPrintJob.Margins property.

#### Properties

```
property Left: Double;
property Top: Double;
property Right: Double;
property Bottom: Double;
```
Left, top, right and bottom margin values.

#### Methods

```
procedure Assign(Source: TPersistent); override;
```
This method copies properties from Source to margins object.

### Other types

```
TDrawTarget = (dtPreview, dtPrint);
```
This type defines draw target: preview or printer. TDrawTarget is used in draw-related methods and events.

```
TUnits = (unPixels, unPercents, unInches, unMillimeters);
```
This type defines units for MarginsUnits, HeaderUnits and FooterUnits of TPrintJob.

```
TDirection = (dirHorizontal, dirVertical);
```
This type defines values for unit conversion direction.

```
TPageMode = (pmDefault, pmCustom);
```
This type defines values for TPrintJob.PageMode property.

```
TPageOrientation = (orDefault, orPortrait, orLandscape);
```
This type defines values for TPrintJob.PageOrientation property.

```
TJobOptions = (joMargins,joHeader,joFooter);
TJobOptionsSet = set of TJobOptions;
```
This types define values for TPrintJob.JobOptions property.

## Sequence of events in TPrintJob

### Print method

```
OnStartPrint
// Printer.BeginDoc call here
  // for each page block start
  OnStartPrintPage
  OnInitDraw
  OnDraw // for page header if joHeader in JobOptions
  OnDraw // for main page area
  OnDraw // for page footer if joFooter in JobOptions
  OnEndPrintPage
  // Printer.NewPage call here
  OnPrintProgress
  // for each page block end
// Printer.EndDoc call here
OnEndPrint
```

### PrintDialog method

(if user close dialog by OK button)

```
OnUpdate
OnPrinterSetupChange
OnStartPrint
// Printer.BeginDoc call here
  // for each page block start
  OnStartPrintPage
  OnInitDraw
  OnDraw // for page header if joHeader in JobOptions
  OnDraw // for main page area
  OnDraw // for page footer if joFooter in JobOptions
  OnEndPrintPage
  // Printer.NewPage call here
  OnPrintProgress
  // for each page block end
// Printer.EndDoc call here
OnEndPrint
```

Events between «for each page block start» and «for each page block end» comments are repeated for each printed page.

## How to realize print job with TPrintJob

To realize print job:
1) Place TPrintJob on your form
2) Set TPrintJob properties as you need
3) Create OnDraw event handler and draw your pages within this event

Example of simpliest realization of OnDraw event:

```
procedure TSomeForm.PrintJobDraw(Sender: TObject;
TheCanvas: TCanvas;
  PageIndex: Integer; Rect: TRect; Area: TDrawArea;
  Target: TDrawTarget);
var
  S: string;
begin
  if Area=daPage then
    with Rect,TheCanvas,Font do
    begin
      // drawing page rectangle
      Rectangle(Left,Top,Right,Bottom);
      // setting font properties
      Name:='Arial';
      Size:=200;
      Style:=[fsBold];
      // drawing text
      S:=IntToStr(PageIndex);
      TextOut(
        (Left+Right-TextWidth(S)) div 2,
        (Top+Bottom-TextHeight(S)) div 2,
        S);
    end;
end;
```

This code draws big centered page number in rectangle.
Full source code is located in
Examples\HowToStart

To print header and footer of the page analyse Area parameter value:

```
case Area of
  daHeader:    // draw page header here
  daPage:      // draw main area of the page here
  daFooter:    // draw page footer here
end;
```

## How to create custom print job class

To create your own print job class:
1) Create new component, derived from TCustomPrintJob class
2) Publish necessary properties and events inherited from TCustomPrintJob
3) Override any virtual methods to create your own functionality
4) Add any methods, properties and events that you need

See example: custom print job component with functionality from example below.

```
type
  TExamplePrintJob = class(TCustomPrintJob)
  private
    { Private declarations }
    // let's add new feature – font choosing
    FFont: TFont;
    procedure SetFont(const Value: TFont);
  protected
    { Protected declarations }
  public
    { Public declarations }
    constructor Create(AOwner: TComponent); override;
    destructor Destroy; override;
    procedure DrawArea(TheCanvas: TCanvas;
      PageIndex: Integer; Rect: TRect;
      Area: TDrawArea; Target: TDrawTarget); override;
  published
    { Published declarations }
    // create new published property
    property Font: TFont read FFont write SetFont;
    // publish only one interited property
    property PageCount;
  end;

. . .

procedure TExamplePrintJob.SetFont(const Value: TFont);
begin
  // realization of the access method for Font property
  FFont.Assign(Value);
  // we must update print job after font changing
  Update;
end;
```

```
constructor TExamplePrintJob.Create(AOwner: TComponent);
begin
  // calling inherited constructor
  inherited;
  // initializing properties
  with Margins do
  begin
    Left:=1;
    Top:=1;
    Right:=1;
    Bottom:=1;
  end;
  MarginsUnits:=unInches;
  Title:='Custom Print Job Demo';
  Options:=[joMargins];
  // creating internal font object
  FFont:=TFont.Create;
  with FFont do
  begin
    // setting default font properties
    Name:='Arial';
    Size:=200;
    Style:=[fsBold];
  end;
end;

destructor TExamplePrintJob.Destroy;
begin
  // destroying font object
  FFont.Free;
  // calling inherited destructor
  inherited;
end;
```

```
procedure TExamplePrintJob.DrawArea(TheCanvas: TCanvas;
  PageIndex: Integer; Rect: TRect;
  Area: TDrawArea; Target: TDrawTarget);
var
  S: string;
begin
  if Area=daPage then
    with Rect,TheCanvas do
    begin
      // drawing page rectangle
      Rectangle(Left,Top,Right,Bottom);
      // assigning Self.Font properties to Canvas.Font
      Font.Assign(FFont);
      // drawing page centered text
      S:=IntToStr(PageIndex);
      TextOut(
        (Left+Right-TextWidth(S)) div 2,
        (Top+Bottom-TextHeight(S)) div 2,
        S);
    end;
end;
```

This example demonstrates creating custom print job with hidden inherited properties and events and some new functionality.
Full source code is located in

Examples\CustomPrintJob

You must install TExamplePrintJob component (ExamplePrintJob.pas) before demo project compiling.

# Print Jobs

! Print Jobs package is available only in Print Suite Pro !

**TCustomGridPrintJob (PJGrid.pas)**
Internal abstract class for realize print of grid. This class defines all based methods and properties for grid printing.

**TCustomSimpleGridPrintJob (PJSimpleGrid.pas)**
Full-functional internal class for grid printing. This class defines events for customize grid and draw grid cells.

**TSimpleGridPrintJob (PJSimpleGrid.pas)**
Published class for grid printing. This class just publishes some properties and events inherited from parent classes.

**TCustomTextGridPrintJob (PJTextGrid.pas)**
Full-functional internal class for realize print of text grid. This class defines properties and methods needed for text grid printing. TCustomTextGridPrintJob class has no published properties and events.

**TTextGridPrintJob (PJTextGrid.pas)**
This full-functional internal class just publish some properties inherited from TCustomPrintJob, TCustomGrid and TCustomTextGridPrintJob.

**TCustomSimpleTextGridPrintJob (PJSimpleTextGrid.pas)**
Full-functional internal class for text grid printing. This class defines events for customize grid and get cells text.

**TSimpleTextGridPrintJob (PJSimpleTextGrid.pas)**
Published class for text grid printing. This class just publishes some properties and events inherited from parent classes.

**TSpecialTextGridPrintJob (PJTextGrid.pas)**
This full-functional internal class is parent for any ready-to-use grid print jobs and overrides DrawArea method to print page number text in footer area.

**TCustomDBGridPrintJob (PJDBGrid.pas)**
Full-functional internal class for realize print of database grid. TCustomDBGridPrintJob class has no published properties and events.

**TDBGridPrintJob (PJDBGrid.pas)**
Published class for database grid printing. This class just publishes some properties and events inherited from parent classes.

**TCustomStringGridPrintJob (PJStringGrid.pas)**
Full-functional internal class for realize print of TStringGrid.
TCustomStringGridPrintJob class has no published properties and events.

**TStringGridPrintJob (PJStringGrid.pas)**
Published class for TStringGrid printing. This class just publishes some properties and events inherited from parent classes.

**TCustomListViewPrintJob (PJListView.pas)**
Full-functional internal class for realize print of TListView.
TCustomListViewPrintJob class has no published properties and events.

**TListViewPrintJob (PJListView.pas)**
Published class for TListView printing. This class just publishes some properties and events inherited from parent classes.

**TCustomStringsPrintJob (PJStrings.pas)**
Full-functional internal class for realize print of TStrings contents (text printing). This class has no published properties and events.

**TStringsPrintJob (PJStrings.pas)**
Published class for text printing. This class just publishes some properties and events inherited from TCustomPrintJob and TCustomStringPrintJob.

**TCustomMultiPrintJob (PJMulti.pas)**
Full-functional internal class for integrating other print jobs. This class has no published properties and events.

**TMultiPrintJob (PJMulti.pas)**
Published class for integrating other print jobs. This class just publishes some events inherited from TCustomPrintJob and TCustomMultiPrintJob.

**TCustomDraftPrintJob (PJDraft.pas)**
Full-functional internal class for draft printing. It allows to place from 4 to 100 draft page of linked print job in one physical page. This class has no published properties and events.

**TDraftPrintJob (PJDraft.pas)**
Published class for draft printing. This class just publishes some events inherited from TCustomPrintJob and TCustomDraftPrintJob.

# TCustomGridPrintJob

TCustomGridPrintJob (PJGrid.pas) is an internal abstract parent class for any grid print jobs. It defines all methods and properties needed for grid print.

## Hierarchy

TObject
    TPersistent
        TComponent
            TCustomPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
```
Standard constructor overriden for initializing properties.

```
procedure DrawCell(TheCanvas: TCanvas;
     Rect: TRect; PageIndex,ACol,ARow: Integer;
     Target: TDrawTarget); virtual; abstract;
```
This virtual method is called when a cell in the grid needs to be drawn. This is an abstract method thus descendant class must override it. Rect parameter contains cell rect, PageIndex contains page number where placed the cell. ACol and ARow parameters contains column and row cell zero-based indexes. In PJGrid.pas unit some special index constants are defined:

`colIndex` grid index column
`rowHeader` grid header row
`rowFooter` grid footer row

Descendant classes use these constants to determine which special cell needs to be drawn.

```
procedure DrawBorders(TheCanvas: TCanvas;
     Rect: TRect; CellType: TCellType; Target:
     TDrawTarget); virtual;
```
This virtual method draws border lines around cell or entire grid. TCellType is defined in PJGrid.pas as:

```
     TCellType = (ctTable,ctHeader,ctPage,ctFooter);
```
Standard classes of Print Jobs don't override this method, but you can override it in your own class to customize cell borders drawing.

```
procedure Draw(TheCanvas: TCanvas; PageIndex: Integer;
     Target: TDrawTarget); override;
procedure DrawArea(TheCanvas: TCanvas;
     PageIndex: Integer; Rect: TRect;
     Area: TDrawArea; Target: TDrawTarget); override;
```
These methods are overriden for grid drawing. You need not to override it in your own grid print classes.

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; virtual; abstract;
```
This virtual method is called to determine column width. This is an abstract method thus descendant class must override it.

**Properties**

```
property ColCount: Integer;
property RowCount: Integer;
```
These properties are the number of columns and rows in the grid.

```
property RowsPerPage: Integer;
```
RowsPerPage is the number of rows are printed in one page. Header and footer rows are not contained in this number.

```
property TableBorders: TCellBorders;
```
This property determines which table borders must be drawn. These borders draw on each page part of grid.

```
property HeaderCellBorders: TCellBorders;
property PageCellBorders: TCellBorders;
property FooterCellBorders: TCellBorders;
```
These properties determine which header, page and footer cell borders must be drawn.

```
property IndexColumn: Boolean;
```
This property enable and disable row index column at the left of the grid. This special column is not contained in ColCount property. Index column are numbered by colIndex constant.

```
property IndexStart: Integer;
```
This property determines base of index column indexing.

These properties are protected in TCustomPrintJob and publish in descendant classes.

# TSimpleGridPrintJob

TSimpleGridPrintJob (PJSimpleGrid.pas) is a component that allows to print any multipage grid by defining only two event handlers: OnColWidth and OnDrawCell.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomGridPrintJob
                    TCustomSimpleGridPrintJob
                        TSimpleGridPrintJob

## Interface

### Methods

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; override;
procedure DrawCell(TheCanvas: TCanvas; Rect: TRect;
     APageIndex,ACol,ARow: Integer;
     Target: TDrawTarget); override;
```

These overriden methods just call OnColWidth and OnDrawCell events.

**Properties**

```
property ColCount;
property RowCount;
property RowsPerPage;
property TableBorders;
property HeaderCellBorders;
property PageCellBorders;
property FooterCellBorders;
property IndexColumn;
property IndexStart;
property Title;
property Margins;
property MarginsUnits;
property Header;
property HeaderUnits;
property Footer;
property FooterUnits;
property PageMode;
property PageWidth;
property PageHeight;
property PageUnits;
property Orientation;
property Options;
```

TSimpleGridPrintJob just publish properties inherited from parent classes. See more information in TCustomGridPrintJob and TCustomPrintJob.

**Events**

```
property OnColWidth: TColWidthEvent;
TColWidthEvent = procedure (Sender: TObject;
     TheCanvas: TCanvas; ACol: Integer;
     var ColWidth: Integer) of object;
```

This event occurs when component needs column width format grid or draw cell. Just analyse ACol parameter and set ColWidth to column width value. If ACol = colIndex width for index column is needed.

```
property OnDrawCell: TDrawCellEvent;
TDrawCellEvent = procedure (Sender: TObject;
     TheCanvas: TCanvas; Rect: TRect;
     PageIndex,ACol,ARow: Integer;
     Target: TDrawTarget) of object;
```

This event occurs when grid cell needs to be drawn. In Rect are passed cell rectangle for area and in ACol and ARow are passed colums and row indexes for cell that must be drawn. Use Target value to determine draw target between printer and preview if you want to create different output (e.g. black and white printing and color preview). If ACol is colIndex cell from index column needs to be drawn. If ARow is rowHeader/rowFooter cell from header/footer needs to be drawn.

# TCustomTextGridPrintJob

TCustomTextGridPrintJob (PJTextGrid.pas) is an internal parent class for any text grid print jobs. It defines all methods and properties needed for text grid print.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomGridPrintJob
                    TCustomTextGridPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal objects.

```
procedure DrawCell(TheCanvas: TCanvas; Rect: TRect;
     APageIndex,ACol,ARow: Integer;
     Target: TDrawTarget); override;
```
This overriden method draw cell text, returned by GetCellText methods with alignment, returned by GetCellAlignment method and using HeaderFont, PageFont and FooterFont.

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; override;
```
This method is intended for getting column width. Result is defined only for ACol=colIndex, otherwise Result is 0. This method is overriden in all descendant classes.

```
function GetCellText(
     APageIndex,ACol,ARow: Integer): string; virtual;
```
This method is intended for getting cell text. Result is defined as

1) for colIndex (‘#’ for rowHeader and row index text for other rows)
2) for last column in the page (page index text)

otherwise Result is ’’;

```
function GetCellAlignment(
     ACol,ARow: Integer): TAlignment; virtual;
```
This method is intended for getting cell text alignment. Result is taRightJustify for colIndex or rowFooter, otherwise Result is taLeftJustify.

**Events**

```
property HeaderFont: TFont;
property PageFont: TFont;
property FooterFont: TFont;
```
These properties contains fonts for drawing header, page and footer cells.

```
property ClipMode: TClipMode;
TClipMode = (cmNone,cmClip,cmEllipsis);
```
This property determines clipping mode for cell text drawing.

`cmNone`

Fastest drawing without text clipping. Use this mode if you are sure that your text does not fall outside the cell.

`cmClip`

In this mode text is clipped on the cell border.

`cmEllipsis`

In this mode part of the text which does not hold in the cell is replaced with ellipses. This is default mode.

# TSimpleTextGridPrintJob

TSimpleGridPrintJob (PJSimpleGrid.pas) is a component that allows to print any multipage text grid by defining only two event handlers: OnColWidth and OnDrawCell.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomGridPrintJob
                    TCustomTextGridPrintJob
                        TCustomSimpleTextGridPrintJob
                            TSimpleTextGridPrintJob

## Interface

### Methods

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; override;
function GetCellText(
     APageIndex,ACol,ARow: Integer): string; override;
function GetCellAlignment(
     ACol,ARow: Integer): TAlignment; override;
```

These overriden methods just call OnColWidth, OnCellText and OnCellAlignment events.

**Properties**

```
property HeaderFont;
property PageFont;
property FooterFont;
property ClipMode;
property ColCount;
property RowCount;
property RowsPerPage;
property TableBorders;
property HeaderCellBorders;
property PageCellBorders;
property FooterCellBorders;
property IndexColumn;
property IndexStart;
property Title;
property Margins;
property MarginsUnits;
property Header;
property HeaderUnits;
property Footer;
property FooterUnits;
property PageMode;
property PageWidth;
property PageHeight;
property PageUnits;
property Orientation;
property Options;
```

Properties inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

**Events**

```
property OnColWidth: TColWidthEvent;
TColWidthEvent = procedure (Sender: TObject;
     TheCanvas: TCanvas; ACol: Integer;
     var ColWidth: Integer) of object;
```

This event occurs when component needs column width format grid or draw cell. Just analyse ACol parameter and set ColWidth to column width value. If ACol = colIndex width for index column is needed.

```
property OnCellText: TCellTextEvent;
TCellTextEvent = procedure (Sender: TObject;
     APageIndex,ACol,ARow: Integer;
     var Text: string) of object;
```
This event occurs when component needs cell text. Just analyse ACol and ARow parameter and set Text to cell text. ARow is a total row index in grid, thus you need use APageIndex only for special cases, e.g. for draw page index in footer or header cells.

```
property OnCellAlign: TCellAlignEvent;
TCellAlignEvent = procedure (Sender: TObject;
     ACol,ARow: Integer;
     var Alignment: TAlignment) of object;
```
This event occurs when component needs cell alignment. Just analyse ACol and ARow parameter and set Alignment to cell alignment.

In PJGrid.pas unit some special index constants are defined:

| | |
|---|---|
| `colIndex` | grid index column |
| `rowHeader` | grid header row |
| `rowFooter` | grid footer row |

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```
Events inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

# TSpecialTextGridPrintJob

TSpecialTextGridPrintJob (PJTextGrid.pas) is a parent component for all ready-to-use grid print jobs.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomGridPrintJob
                    TCustomTextGridPrintJob
                        TSpecialTextGridPrintJob

## Interface

### Methods

```
procedure DrawArea(TheCanvas: TCanvas;
     PageIndex: Integer; Rect: TRect;
     Area: TDrawArea; Target: TDrawTarget); override;
```

TSpecialTextGridPrintJob overrides this method to draw page index information in page footer in all ready-to-use descendants.

# TStringGridPrintJob

TStringGridPrintJob (PJStringGrid.pas) is a ready-to-use component for easy TStringGrid print. You need not to write manual code for printing. Component gets all data for printing, including row and column counts and cell texts from linked TStringGrid.

## Hierarhy

TObject
TPersistent
TComponent
TCustomPrintJob
TCustomGridPrintJob
TCustomTextGridPrintJob
TSpecialTextGridPrintJob
TCustomStringGridPrintJob
TStringGridPrintJob

## Interface

### Methods

```
procedure Update; override;
```
This overriden method gets column and row counts from linked TStringGrid then calls inherited Update.

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; override;
```
This overriden method calculates column width as proportional part of page width using column width and sum of column widths of linked TStringGrid.

```
function GetCellText(
     APageIndex,ACol,ARow: Integer): string; override;
```
This overriden method returns text of TStringGrid.Cells[ACol,ARow].

**Notes.**
If Options contains joHeader first row in TStringGrid are interpreted as grid header. PageCount is calculated automatically using RowsPerPage and RowCount, you cannot access to this property.

**Properties**

```
property StringGrid: TStringGrid;
```
This property contains link to associated TStringGrid.

```
property HeaderFont;
property PageFont;
property FooterFont;
property ClipMode;
property ColCount;
property RowCount;
property RowsPerPage;
property TableBorders;
property HeaderCellBorders;
property PageCellBorders;
property FooterCellBorders;
property IndexColumn;
property IndexStart;
property Title;
property Margins;
property MarginsUnits;
property Header;
property HeaderUnits;
property Footer;
property FooterUnits;
property PageMode;
property PageWidth;
property PageHeight;
property PageUnits;
property Orientation;
property Options;
```
Properties inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

**Events**

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```
Events inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

# TDBGridPrintJob

TDBGridPrintJob (PJDBGrid.pas) is a ready-to-use component for easy database grid print. You need not to write manual code for printing. Component gets all data for printing, including row and column counts, cell texts and cell alignment from linked data set through special TDataSource descendant.

## Hierarhy

TObject
TPersistent
TComponent
TCustomPrintJob
TCustomGridPrintJob
TCustomTextGridPrintJob
TSpecialTextGridPrintJob
TCustomDBGridPrintJob
TDBGridPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal objects.

```
procedure Update; override;
```
This overriden method gets column and row counts from linked data set then calls inherited Update. RowCount determines as DataSource.DataSet.RecordCount and ColCount determines as number of visible fields of DataSource.DataSet.

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; override;
```
This overriden method calculates column width as proportional part of page width using DisplayWidth of the column and sum of DisplayWidth values of visible columns of DataSource.DataSet.

```
function GetCellText(
     APageIndex,ACol,ARow: Integer): string; override;
```
This overriden method returns text of data set field for record numbered as ARow or DisplayName of appropriate data set field.

```
function GetCellAlignment(
     ACol,ARow: Integer): TAlignment; override;
```
This overriden method returns alignment using Alignment property of appropriate data set column.

**Notes.**
ACol does not correspond to Fields array index because ACol is index for visible columns only.
To customize this print job just edit fields information in association data set (TTable, TSQL, etc.). Use “Fields editor” from data set component popup menu in design-time then change Visible, DisplayWidth, DisplayName and Alignment properties of data set fields or change properties of Fields array items in run-time.

**Properties**

```
property DataSource: TDataSource;
```
This property contains link to associated TDataSource to link with data set.

```
property HeaderFont;
property PageFont;
property FooterFont;
property ClipMode;
property ColCount;
property RowCount;
property RowsPerPage;
property TableBorders;
property HeaderCellBorders;
property PageCellBorders;
property FooterCellBorders;
property IndexColumn;
property IndexStart;
property Title;
property Margins;
property MarginsUnits;
property Header;
property HeaderUnits;
property Footer;
property FooterUnits;
property PageMode;
property PageWidth;
property PageHeight;
property PageUnits;
property Orientation;
property Options;
```
Properties inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

**Events**

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```

Events inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

# TListViewPrintJob

TListViewPrintJob (PJListView.pas) is a ready-to-use component for easy TListView contents print. You need not to write manual code for printing. Component gets all data for printing, including row and column counts, cell texts and icons from linked TListView.

## Hierarhy

TObject
TPersistent
TComponent
TCustomPrintJob
TCustomGridPrintJob
TCustomTextGridPrintJob
TSpecialTextGridPrintJob
TCustomListViewPrintJob
TListViewPrintJob

## Interface

### Methods

```
procedure Update; override;
```
This overriden method gets column and row counts from linked TListView then calls inherited Update.

```
procedure DrawCell(TheCanvas: TCanvas;
     Rect: TRect; PageIndex,ACol,ARow: Integer;
     Target: TDrawTarget); override;
```
This overriden method draws icon and text if ACol=0 otherwise calls inherited DrawCell.

```
function GetColWidth(TheCanvas: TCanvas;
     ACol: Integer): Integer; override;
```
This overriden method calculates column width as proportional part of page width using column width and sum of column widths of linked TListView.

```
function GetCellText(
     APageIndex,ACol,ARow: Integer): string; override;
```
This overriden method returns text caption for ACol=0 otherwise SubItems[Pred(ACol)] of appropriate list item. If ARow is rowHeader list view column captions are returned.

```
function GetCellAlignment(
     ACol,ARow: Integer): TAlignment; override;
```
This overriden method returns Alignment of appropriate list view column.

**Properties**

```
property ListView: TListView;
```
This property contains link to associated TListView.

```
property DrawIcons: Boolean;
```
If this property set to True icons are drawn in first column. If associated list view has assigned LargeIcons properties large icon is drawn, otherwise small icon.

```
property HeaderFont;
property PageFont;
property FooterFont;
property ClipMode;
property ColCount;
property RowCount;
property RowsPerPage;
property TableBorders;
property HeaderCellBorders;
property PageCellBorders;
property FooterCellBorders;
property IndexColumn;
property IndexStart;
property Title;
property Margins;
property MarginsUnits;
property Header;
property HeaderUnits;
property Footer;
property FooterUnits;
property PageMode;
property PageWidth;
property PageHeight;
property PageUnits;
property Orientation;
property Options;
```
Properties inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

**Events**

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```

Events inherited from parent classes. See TCustomTextGridPrintJob, TCustomGridPrintJob and TCustomPrintJob.

# TStringsPrintJob

TStringsPrintJob (PJStrings.pas) is a component for easy text files and TStrings contents print. You need not to write manual code for printing.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomStringsPrintJob
                    TStringsPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal objects.

```
procedure Update; override;
```
This method is overriden for calculating PageCount and internal fields using Font parameters and Strings contents.

```
procedure DrawArea(TheCanvas: TCanvas;
     PageIndex: Integer; Rect: TRect;
     Area: TDrawArea; Target: TDrawTarget); override;
```
This method is overriden for text drawing.

**Properties**

```
property Strings: TStrings;
```
This property contains text for printing.

```
property Font: TFont;
```
This property contains Font for printing.

```
property Title;
property Margins;
property MarginsUnits;
property Header;
property HeaderUnits;
property Footer;
property FooterUnits;
property PageMode;
property PageWidth;
property PageHeight;
property PageUnits;
property Orientation;
property Options;
```
Properties inherited from TCustomPrintJob.

**Events**

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```
Events inherited from TCustomPrintJob.

# TMultiPrintJob

TMultiPrintJob (PJMulti.pas) is a component for integrate other print jobs. You need not to write manual code for printing. All integrated print jobs are stored in internal list. You can access this list by public methods and properties in run-time and by special editor in design-time. To call the editor choose “Print Jobs editor…” from context menu of the component or by double-click on the component.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomMultiPrintJob
                    TMultiPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal objects.

```
procedure Update; override;
```
This method is overriden for calculating PageCount as sum of PageCount of integrated print jobs.

```
procedure Draw(TheCanvas: TCanvas; PageIndex: Integer;
     Target: TDrawTarget); override;
```
This overriden metod determines the print job and local page index from PageIndex and calls Draw method for appropriate print job.

```
procedure Add(APrintJob: TCustomPrintJob);
```
This method adds print job into print job list.

```
procedure Insert(Index: Integer;
     APrintJob: TCustomPrintJob);
```
This method inserts print job into print job list at a specified position.

```
procedure Clear;
```
This method empties print job list.

```
procedure Delete(Index: Integer);
```
This method deletes print job at a specified position from print job list.

```
function IndexOf(APrintJob: TCustomPrintJob): Integer;
```
This method returns the position of specified print job in the print job list.

```
function IndexOfName(AName: string): Integer;
```
This method returns the position of print job specified by print job component name in the print job list.

```
procedure Move(CurIndex, NewIndex: Integer);
```
This method changes the position of a print job in print job list.

Methods ahead are intended for editing print job list in run-time.

```
function GetPageJob(Page: Integer): TCustomPrintJob;
```
This method returns page job calculated from global page index.

```
function GetLocalPage(Page: Integer): Integer;
```
This method translates global page index into local page index of one of print jobs in print job list. To get print job from global page index use GetPageJob method.

```
procedure EditPrintJobs;
```
This method calls standard print job list editor.

**Properties**

**published**

```
property Title;
```
Property inherited from TCustomPrintJob.

**public**

```
property PrintJobCount: Integer;
```
This read-only property contains count of print jobs in print job list.

```
property PrintJobs[Index: Integer]: TCustomPrintJob;
```
This read-only property contains all integrated print jobs.

**Events**

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```

Events inherited from TCustomPrintJob.

# TDraftPrintJob

TDraftPrintJob (PJDraft.pas) is a component for draft printing. You need not to write manual code for it, just link component with another print job.

## Hierarhy

TObject
    TPersistent
        TComponent
            TCustomPrintJob
                TCustomDraftPrintJob
                    TDraftPrintJob

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
```
This is standard constructor, overridden for initializing properties.

```
procedure Update; override;
```
This method is overriden for calculating PageCount and copying properties values from linked print job.

```
procedure Draw(TheCanvas: TCanvas; PageIndex: Integer;
     Target: TDrawTarget); override;
```
This overriden method scales TheCanvas according Scale value and draws draft pages.

```
procedure DrawSeparator(TheCanvas: TCanvas;
     FromPoint,ToPoint: TPoint); virtual;
```
This virtual method draws separator lines between draft pages. By default, dotted black line is used. You can override this method to realize your own seperator style.

### Properties

```
property PrintJob: TCustomPrintJob;
```
This property contains link to another print job.

```
property Scale: Integer;
```
This property contains draft scale. Value of this property may be between 2 and 10. Draft page count, placed on one physical page is square of Scale, thus draft page count may be between 4 and 100 on one physical page.

```
property Separators: Boolean;
```
If this property set to True, separator lines is drawn between draft pages.

```
property Title;
```
Property inherited from TCustomPrintJob.

**Events**

```
property OnCreate;
property OnDestroy;
property OnPrinterSetupChange;
property OnStartPrint;
property OnEndPrint;
property OnPrintProgress;
property OnStartPrintPage;
property OnEndPrintPage;
property OnUpdate;
```
Events inherited from TCustomPrintJob.

# TPreview

TPreview (PSPreview.pas) is a visual component for simple creating preview for print job. Just place TPreview on your form and link component with TPrintJob object by PrintJob property. TPreview allows you to change view mode and view scale, scroll pages, views margins, etc.

## Hierarchy

TObject
    TPersistent
        TComponent
            TControl
                TWinControl

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal classes.

```
procedure Update; override;
```
This method update TPreview and all linked controls.

```
procedure AddControlNotification(Control: TControl);
procedure DeleteControlNotification(Control: TControl);
```
These are internal methods to link and unlink controls with TPreview. All additional preview controls (TPreviewToolbar, TPreviewStatusBar, etc.) in Print Suite call these methods when their Preview property is changed.

```
procedure ZoomIn;
procedure ZoomOut;
```
These methods zoom in and out preview to nearest fixed scale value. To change fixed scale values see «Customization and localization».

```
function DPIX: Integer;
function DPIY: Integer;
```
These functions return horizontal and vertical resolution of preview current view scale adjusted. Don’t miss this functions with similar functions of TPrintJob

**Properties**

**public**

```
property ScaleText: string;
```
This read-only property contains information about current preview scale and mode in text format. To change format see «Customization and localization».

```
property PageText: string;
```
This read-only property contains information about current page index and page count in text format. To change format see «Customization and localization».

**published**

```
property PrintJob: TCustomPrintJob;
```
This property links TPreview with TPrintJob object.

```
property ViewMode: TViewMode;
TViewMode = (vmWholePage, vmPageWidth, vmCustom);
```
View mode. Change this property to switch between custom scale, viewing whole page and scaling by page width.

```
property ViewScale: Integer;
```
View scale. When ViewMode is wmWholePage or wmPageWidth ViewScale is read-only and contains current scale for these modes otherwise you can set any view scale value between 10% and 1000%. To change this min and max values see «Customization and localization».

```
property MarginsVisible: Boolean;
```
Set this property to True if you want to display margins in preview.

```
property MarginsFill: TColor;
```
Fill color for margins area.

```
property MarginsBorder: TColor;
```
Color for border of margins area.

```
property PageIndex: Integer;
```
Current page index. Value may be between 1 and PrintJob.PageCount.

```
property ScrollTracking: Boolean;
```
If value of this property is set to True, the preview moves as you drag the thumb tab. Otherwise preview remains static until the thumb tab is released.

```
property Align;
property BorderStyle;
property Ctl3D;
property Cursor;
property ParentCtl3D;
property TabStop;
property TabOrder;
property Visible;
```
Standard properties.

**Events**

```
property OnUpdate: TNotifyEvent;
```
This event occurs after TPreview update.

```
property OnPageChanged: TNotifyEvent;
```
This event occurs after current page index change.

```
property OnEnter;
property OnExit;
property OnKeyDown;
property OnKeyPress;
property OnKeyUp;
```
Standard events.

# TPreviewToolbar

TPreviewToolbar (PSToolbar.pas) is a visual component for control of TPreview parameters.

## Hierarchy

TObject
    TPersistent
        TComponent
            TControl
                TWinControl
                    TCustomControl
                        TCustomPanel

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal classes.

```
procedure Update; override;
```
This method updates TPreviewToolbar and contained controls.

```
procedure UpdateButtons;
procedure EnableButtons;
```
These internal methods update contained speed buttons.

```
procedure ButtonClick(Button: TButtonKind); virtual;
TButtonKind = (btnPrint, btnPrintDialog,
btnPrinterSetupDialog,
  btnFirstPage, btnPrevPage, btnNextPage, btnLastPage,
btnZoomOut,
  btnZoomIn, btnRefresh, btnClose, btnUser);
```
This virtual method processes clicks of toolbar buttons.

**Properties**

```
property Preview: TPreview;
```
This property links toolbar with TPreview object.

```
property Buttons: TButtons;
TButtons = set of TButtonKind;
```
This property determines visibility of toolbar buttons.

```
property Flat: Boolean;
```
If you set this property value to True all toolbar buttons will be flat otherwise all buttons will have 3D border.

```
property ScaleBox: Boolean;
```
Scale combo box visibility.

```
property ScaleBoxPos: TButtonKind;
```
Position for scale combo box. You can specify the button which will be placed into the combo box.

```
property UserButtonEnabled: Boolean;
```
Use this property to enable and disable user button.

```
property UserButtonGlyph: TBitmap;
```
Image for user button. Recommended size is 16 x 16 pixels.

```
property UserButtonHint: string;
```
Hint for user button.

```
property Align;
property ParentShowHint;
property ShowHint;
property Visible;
```
Standard properties.

**Events**

```
property OnUpdate: TNotifyEvent;
```
This event occurs after toolbar update.

```
property OnUserButtonClick: TNotifyEvent;
```
This event occurs when user clicks user button.

# TPreviewStatusBar

TPreviewStatusBar (PSStatusBar.pas) is a visual component for displaying view scale/mode and page index of TPreview component.

## Hierarchy

TObject
    TPersistent
        TComponent
            TControl
                TWinControl

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying internal classes.

```
procedure Update; override;
```
This method updates TPreviewStatusBar.

### Properties

```
property Preview: TPreview;
```
This property links status bar with TPreview object.

```
property Panels: TPanels;
TPanelKind = (pnPage, pnScale);
TPanels = set of TPanelKind;
```
This property show and hide special preview information panels.

```
property PagePanelWidth: Integer;
```
Use this property to change width of page index information panel.

```
property ScalePanelWidth: Integer;
```
Use this property to change width of page scale information panel.

```
property UserText: string;
```
Use this property to set text of third panel.

```
property SizeGrip: Boolean;
```
If this property is set to True, the status bar has a triangular grip on the lower right corner.

```
property Font;
property ParentFont;
property ParentShowHint;
property PopupMenu;
property ShowHint;
property Visible;
```
Standard properties.

**Events**

```
property OnUpdate: TNotifyEvent;
```
This event occurs after status bar update.

# TPreviewComboBox

TPreviewComboBox (PSCombo.pas) is a visual component for choosing and displaying view scale and view mode of TPreview component.

## Hierarchy

TObject
    TPersistent
        TComponent
            TControl
                TWinControl
                    TCustomComboBox

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties.

```
procedure Update; override;
```
This method updates TPreviewComboBox.

### Properties

```
property Preview: TPreview;
```
This property links combo box with TPreview object.

```
property Color;
property Ctl3D;
property Font;
property ParentColor;
property ParentCtl3D;
property ParentFont;
property ParentShowHint;
property PopupMenu;
property ShowHint;
property TabOrder;
property Visible;
```
Standard properties.

**Events**

```
property OnUpdate: TNotifyEvent;
```
This event occurs after combo box update.

# TPreviewLabel

TPreviewLabel (PSLabel.pas) is a visual component displaying view scale or view mode of TPreview component.

## Hierarchy

TObject
    TPersistent
        TComponent
            TControl
                TGraphicControl
                    TCustomLabel

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties.

```
procedure Update; override;
```
This method updates TPreviewLabel.

### Properties

```
property Preview: TPreview;
```
This property links label with TPreview object.

```
property Mode: TMode;
TMode = (lmPage, lmScale);
```
Use this property to choose display mode. TPreviewLabel can display scale or page index information.

```
property Align;
property Alignment;
property AutoSize;
property Color;
property Font;
property Layout;
property ParentColor;
property ParentFont;
property ParentShowHint;
property PopupMenu;
property ShowHint;
property Transparent;
property Visible;
```
Standard properties.

**Events**

```
property OnUpdate: TNotifyEvent;
```
This event occurs after combo box update.

# TPreviewWindow

TPreviewWindow (PSWindow.pas) is a non-visual component for creating preview window. Component can show preview window in modal and non-modal modes.

## Hierarchy

TObject
    TPersistent
        TComponent

## Interface

### Methods

```
constructor Create(AOwner: TComponent); override;
destructor Destroy; override;
```
These are standard constructor and destructor, overridden for initializing properties and creating/destroying contained controls.

```
procedure Update; override;
```
This method updates TPreviewWindow.

```
procedure Show;
```
Call this method to show preview window.

```
procedure ShowModal;
```
Call this method to show preview window as modal window.

```
procedure Close;
```
Call this method to close preview window.

**Properties**

```
property PrintJob: TCustomPrintJob;
```
This property links TPreviewWindow with TPrintJob object.

```
property BorderIcons: TBorderIcons;
property BorderStyle: TFormBorderStyle;
property Caption: string;
property Ctl3D: Boolean;
property Font: TFont;
property Icon: TIcon;
property Position: TPosition;
property WindowState: TWindowState;
property Width: Integer;
property Height: Integer;
```
Form properties. See TForm to get more information.

```
property MarginsBorder: TColor;
property MarginsFill: TColor;
property MarginsVisible: Boolean;
property PageIndex: Integer;
property ScrollTracking: Boolean;
property ViewMode: TViewMode;
property ViewScale: Integer;
```
Contained TPreview object properties. See TPreview to get more information.

```
property Buttons: TButtons;
property Flat: Boolean;
property ScaleBox: Boolean;
property ScaleBoxPos: TButtonKind;
property ShowHint: Boolean;
property UserButtonEnabled: Boolean;
property UserButtonGlyph: TBitmap;
property UserButtonHint: string;
```
Contained TPreviewToolbar object properties. See TPreviewToolbar to get more information.

```
property Panels: TPanels;
property PagePanelWidth: Integer;
property ScalePanelWidth: Integer;
property UserText: string;
property SizeGrip: Boolean;
```
Contained TPreviewStatusBar object properties. See TPreviewStatusBar to get more information.

**Events**

```
property OnCreate: TNotifyEvent;
property OnDestroy: TNotifyEvent;
```
These events occur when component are creating and destroying.

```
property OnShow: TNotifyEvent;
```
This event occurs before preview window open.

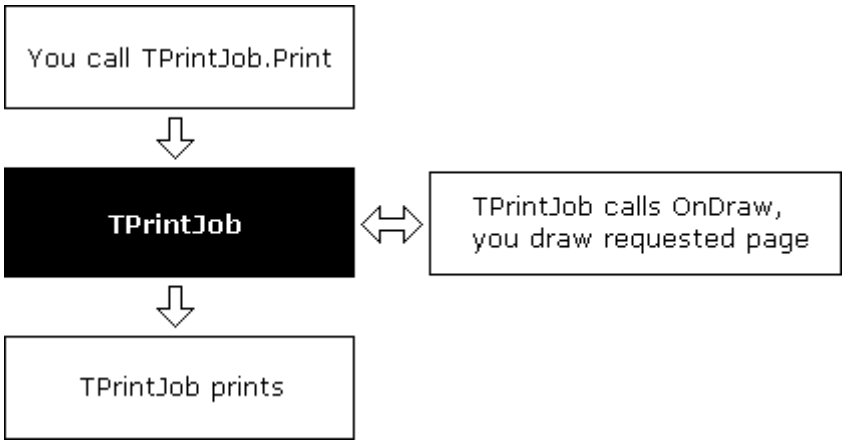
```
property OnClose: TNotifyEvent;
```
This event occurs before preview window close.

```
property OnUpdate: TNotifyEvent;
property OnPageChanged: TNotifyEvent;
```
Contained TPreview object events. See TPreview to get more information.

# How it works?

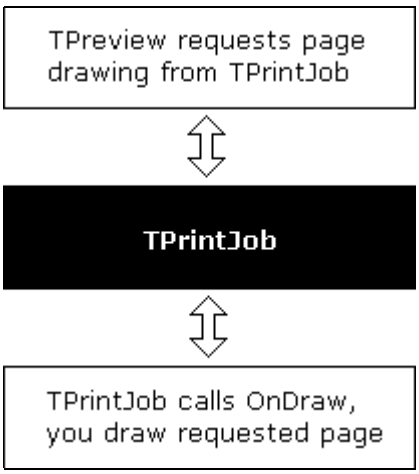
## How TPrintJob works?

TPrintJob knows which page must be drawn when either printing or previewing, and calls OnDraw event for the page. Parameters are passed to OnDraw, including: canvas for drawing, page index, page area, and even the draw target. Just respond to OnDraw event and create graphics for each page.




## How TPreview works?

TPreview requests drawing of the current displayed page from TPrintJob. TPrintJob calls OnDraw event; you draw requested page, and your graphics are shown in preview.



# Customization and localization

All common constants is defined in PSCommon.pas module. You can change these constants to customize and/or localize your Print Suite. PSCommon.pas contains next constants:

```
ScaleArray: array[0..10] of Integer =
(10,25,50,75,100,150,200,300,500,700,1000);
```

This array contains series of integer values, used in TPreviewCombo box to creating drop-down list items and in TPreview to determine previous and next scale for ZoomIn and ZoomOut methods. You can change both and dimension of this array and items value. First and last items are used as minimal and maximal scale values.

```
strScaleComboHint = 'View mode and scale';
```

This constant contains text assigned to Hint property of TPreviewComboBox in TPreviewToolbar. Change this string value to localize your Print Suite.

```
HintArray: array[0..10] of string = (
  'Print',
  'Print dialog',
  'Printer setup dialog',
  'First page',
  'Previous page',
  'Next page',
  'Last page',
  'Zoom out',
  'Zoom in',
  'Refresh',
  'Close');
```

This array contains strings that used as hints of buttons in TPreviewToolbar. You can change only values, not dimension of array.

```
strNotLinked = 'not linked';
```

This constant contains text for display in non-linked preview controls. Choose this text to localize your Print Suite.

```
strPageWidth = 'Page width (%d%%)';
strWholePage = 'Whole page (%d%%)';
strPageIndex = 'Page %d from %d';
```

These constants contain texts that are used in ScaleText and PageText of TPreview. Choose this texts to localize your Print Suite.

```
strComboPageWidth = 'Page width';
strComboWholePage = 'Whole page';
```

These constants contain texts that used in TPreviewComboBox to fill drop-down list. Choose this texts to localize your Print Suite.

```
strComboInvalidScale = 'Invalid scale value';
```
This constant contains text for display error when invalid scale value is entered. Choose this text to localize your Print Suite.

# Demo and examples

All main features of Greatis Print Suite and Greatis Print Suite Pro are demonstrated in special demo project and examples. Compiled executable file with source is available in free demo package, separated examples are available after source codes purchase.

## Parts of demo project

### About Greatis Print Suite

This part just displays information about Greatis Print Suite and prints simpliest two-page graphics with name, slogan, home page link and copyrights of Print Suite.

### Print job page areas

This part demonstrates areas of print job page. Simple code fills page areas with different colors:

```
procedure TfrmMain.psjAreasDraw(Sender: TObject;
  TheCanvas: TCanvas; PageIndex: Integer;
  Rect: TRect; Area: TDrawArea; Target: TDrawTarget);
begin
  with TheCanvas,Font,Rect do
  begin
    with Brush do
      case Area of
        daHeader: Color:=$CCFFCC;
        daPage: Color:=$CCCCFF;
        daFooter: Color:=$FFCCCC;
      end;
    FillRect(Rect);
  end;
end;
```

You can choose any Header, Footer and Margins parameters and see how they change page.

### Published interface of Print Suite components

One more example of simple print job realization. Thirteen pages of this example shows published interfaces of Greatis Print Suite Pro components.

**Simpliest realization of print job**

This example demonstrates main idea of Print Suite: with TPrintJob you must draw only requested page:

```
procedure TfrmMain.psjSimpliestDraw(Sender: TObject;
  TheCanvas: TCanvas; PageIndex: Integer; Rect: TRect;
  Area: TDrawArea; Target: TDrawTarget);
var
  S: string;
begin
  if Area=daPage then
    with Rect,TheCanvas,Font do
    begin
      // drawing page rectangle
      Rectangle(Left,Top,Right,Bottom);
      // setting font properties
      Name:='Arial';
      Size:=200;
      Style:=[fsBold];
      // drawing text
      S:=IntToStr(PageIndex);
      TextOut(
        (Left+Right-TextWidth(S)) div 2,
        (Top+Bottom-TextHeight(S)) div 2,
        S);
    end;
end;
```

In source code kit this example is realized in two variants: by OnDraw event and by creating special print job class. These variants are located in Examples\HowToStart\ and Examples\CustomPrintJob\.

**Exact size graphic printing**

This example demonstrates how to print exact size graphic objects. Accuracy of rules printing in this example is depended from printer resolution only. To calculating pixel sizes special methods of TPrintJobs are used.

```
LB:=Round(InchToPix(0.3,dirVertical));
LM:=Round(0.75*InchToPix(0.3,dirVertical));
LS:=Round(0.5*InchToPix(0.3,dirVertical));
```

**Database detail printing – one record in one page**

This example demonstrates simpliest printing detailed database report using standard TPrintJob component. Before getting and printing table record contents you must to position table by simple code:

```
procedure TfrmMain.psjDBDetailDraw(Sender: TObject;
  TheCanvas: TCanvas; PageIndex: Integer; Rect: TRect;
  Area: TDrawArea; Target: TDrawTarget);
...
begin
  Table.RecNo:=PageIndex;
...
```

Separated example is available in Examples\Database\.

**Database printing in grid mode (Pro only)**

This example demonstrates using of TDBGridPrintJob component. You don’t need to write manual code to print database table, just link TDBGridPrintJob with DataSource and customize fields of dataset. TDBGridPrintJob gets information about fields (width, caption, text alignment) from Fields array of dataset. Separated example is available in Examples\PrintJobs\SimpleTextGrid.

**Simpliest text grid printing (Pro only)**

This example demonstrates using of TSimpleTextGridPrintJob – special component for easiest printing text tables. Just create handlers for OnColWidth, OnCellAlign and OnCellText events:

```
procedure TfrmMain.psjSimpleTextGridColWidth(Sender:
TObject;
  TheCanvas: TCanvas; ACol: Integer; var ColWidth:
Integer);
begin
  with Sender as TSimpleTextGridPrintJob,GetPageRect do
    ColWidth:=(Right-Left) div ColCount;
end;
```

```
procedure TfrmMain.psjSimpleTextGridCellText(Sender:
TObject; APageIndex,
  ACol, ARow: Integer; var Text: String);
begin
  case ARow of
    rowHeader: Text:=Format('H.%d',[ACol]);
    rowFooter: Text:=Format('F.%d',[ACol]);
  else Text:=Format('%d.%d',[ACol,ARow]);
  end;
end;
```

Separated example is available in Examples\PrintJobs\SimpleTextGrid\.

**Simpliest graphic grid printing (Pro only)**

This example demonstrates using of TSimpleGridPrintJob – special component for easiest graphic text tables. Just create handlers for OnColWidth and OnDrawCell events. Separated example is available in Examples\PrintJobs\SimpleGrid\.

**TStringGrid contents printing (Pro only)**

This example demonstrates using of TStringGridPrintJob component. You don’t need to write manual code to print grid contents, just link TStringGridPrintJob with TStringGrid. TStringGridPrintJob calculates column widths from relative column widths of TStringGrid. Separated example is available in Examples\PrintJobs\StringGrid\.

**TListView contents printing (Pro only)**

This example demonstrates using of TListViewPrintJob component. You don’t need to write manual code to print list view, just link TListViewPrintJob with TListView. TListViewPrintJob gets information about columns properties (caption, width, alignment) from Columns property of TListView. Separated example is available in Examples\PrintJobs\ListView\.

**Notes.**
All grid print jobs have separated font properties for header, footer and main area of the page.
All grid print jobs have properties for controlling cell and table borders drawing.

### TStrings printing (Pro only)

This example demonstrates using of TStringsPrintJob component. You don’t need to write manual code to print TStrings contents, just put your text into Strings property. This component can be used for text file printing. To print text file load text into Strings property by calling Strings.LoadFromFile method. Separated example is available in Examples\PrintJobs\Strings\.

### Multi-job printing (Pro only)

This example demonstrates using of TMultiPrintJob. This component allows to integrate any print jobs into one print job. To use just add print jobs to PrintJobs list.

### Draft printing (Pro only)

This example demonstrates using of TDraftPrintJob. This component allows to place up to 100 draft page on one physical page. To use just link TDraftPrintJob with another print job.

# Contact information

**Greatis Print Suite / Greatis Print Suite Pro**

| | |
|---|---|
| Home page: | www.greatis.com/printsuite.htm |
| Newest documentation | www.greatis.com/printsuite.pdf |
| Free demo | www.greatis.com/printsuitedemo.zip |
| Order page | www.greatis.com/printsuitebuy.htm |
| Support e-mail: | b-team@greatis.com |

**Greatis Software**

| | |
|---|---|
| Home page: | www.greatis.com |