MacApp® 2.0b5 Printing Release Notes

Curt Bianchi

This ERS is separated into three parts. The first part is a general overview of how printing works in MacApp. I felt this was necessary since the original MacApp manual had very limited documentation on printing. The second section describes the view methods related to printing, and the third section describes the `TStdPrintHandler` class.

How Printing Works in MacApp

Printing in MacApp is accomplished by objects called *print handlers*. MacApp supplies two print handler classes. `TPrintHandler` is a "null" print handler that isn't capable of printing: it simply defines the minimal print handler interface. `TStdPrintHandler`, in the UPrinting unit, fully implements standard Macintosh printing for spooled (ImageWriter®) and non-spooled (LaserWriter®) devices, as well as handling printing-related issues such as page setup and screen feedback of printing information.

Printing is accomplished through the cooperation of a print handler and a view. A print handler must always be coupled to a view; otherwise the print handler has nothing to print. Print handlers handle all of the mechanics of printing (the print loop, spooling, communicating with the Print Manager, and so on), while the view performs the actual drawing within each page. A set of view methods exists that are called from the print handler. These methods allow you to customize many aspects of printing for a particular view, without the need to override `TStdPrintHandler`. To customize print behavior it is usually sufficient to override the view methods related to printing. It is rare that `TStdPrintHandler` must be overridden, but you certainly aren't prevented from doing so.

A print handler is usually created along with the view it prints, in the document's `DoMakeViews` method. Only one print handler needs to be created for a hierarchy of views. The print handler will print the view with which it is associated and all of its subviews.

Types of Printing Setup Supported by TStdPrintHandler

Printing is performed by calling the print handler's `Print` method. Generally you don't call `Print` directly. It is called for you by the print handler's `DoMenuCommand`, in response to Print or Print One commands, or when Finder printing. MacApp will perform different types of setup before calling `Print`. The types of setup are as follows:

Normal	Before `Print` is called, `PoseJobDialog` is called to gather print job information. This is the technique used when the user chooses the Print menu command.
Print One	No job dialog is displayed. The print job information is set up by validating the print record and forcing all pages to be printed. Therefore, any valid print record information will be carried forward from the last time the print handler printed. This is the technique used when the user chooses the Print One menu command.

Finder	The page setup and job dialogs may be posed before printing. The use of the dialogs is controlled by the `TStdPrintHandler` flags `fFinderPageSetup` and `fFinderJobDialog`. The job dialog will be displayed at least once, even if `fFinderJobDialog` is false. For each subsequent document the job information is forwarded to each job via `PrJobMerge`. This is the technique used for Finder printing.

You may find wish to implement a nonstandard print command, in which case you might choose a different method of setup, and then call `Print` yourself.

Page Definitions

The dimensions of a physical page are kept in a print handler field called `fPageAreas`. It is a record containing the following information:

`theInk`	A rectangle defining the printable part of the page. Its top-left corner is always (0, 0). Its bottom-right corner is the maximum printable page height and width attainable given the current printer setup.
`thePaper`	A rectangle defining the entire physical page, in the coordinate system of `theInk`. Since the physical page is typically larger than the printable part of the page, the top-left corner of `thePaper` is negative and the bottom-right coordinates are greater than those of `theInk`.
`theInterior`	A rectangle defining the part of the page in which the print handler draws its view, in the coordinate system of `theInk`. `theInterior` is never larger than `theInk`. It may be smaller than `theInk` if the application wishes to impose page margins.
`theMargins`	A rectangle defining the top, left, bottom, and right paper margins. The margins define the distance from the edge of the physical page to the interior of the page. The margins can never be smaller than the difference between `theInk` and `thePaper`. `theInterior` is computed by subtracting `theMargins` from `thePaper`.

The page areas are recomputed whenever the user chooses a new printer. Furthermore `theMargins` (and `theInterior`) may be changed by calling the print handler method `InstallMargins`. To better understand how these fields are related, run a MacApp sample program and inspect its `TStdPrintHandler` objects.

Page Strips and Page Breaks

The view associated with a print handler may specify what part of its extent is to be printed. Typically, it is equivalent to the view's entire extent. Since the extent of the view is usually larger than a page, it is necessary to divide the view into pages. Typically, we think of pages as being arranged vertically, but it is also possible to have pages arranged horizontally if the view's width is greater than the width of a page. Thus you can think of the view as being divided into rows and columns of pages. Rather then use the terms "row" and "column," we use the term *page strip*. Vertical page strips are analogous to columns: the number of vertical page strips in a view is the number of "columns" of pages; horizontal page strips are analogous to rows. This may be counter-intuitive at first, so when you think of page strips think of rows and columns.

The size of the horizontal and vertical page strips may be uniform or variable. Page strips of varying sizes are used by views that print a different amount of themselves on each page. `TTEView` uses this feature to ensure that page breaks occur on line boundries. The view method `DoCalcViewPerPage` determines the standard page strip size. Unless overridden, it calls the print handler method `CalcViewPerPage`, which simply returns the size of the page interior.

The number of page strips is determined by calling the view's `DoCalcPageStrips` method. Unless overridden, it in turn calls the print handler method `CalcPageStrips`. This method uses one of two techniques to determine the number of page strips in each direction: If the page strip size is fixed, then the number of page strips is computed by dividing the view's height (width) by the page strip height (width). If the page strip size is not fixed, then `CalcPageStrips` computes each and every page break, counting them up as it goes.

To determine where a page break occurs, the view method `DoBreakFollowing` is called. The purpose of `DoBreakFollowing` is to return the view coordinate at which a page break occurs after the page break previously computed. Unless overridden, it calls the print handler method `BreakFollowing`, which simply adds the strip size to the last page break. To implement a view whose page strips vary in size, you must override `TView.DoBreakFollowing`.

Two other print handler methods are closely related to page breaks. The method `TStdPrintHandler.EachBreak` is used to perform some function on each page break in the view. `TStdPrintHandler` uses it to compute the number of page strips for variable size strips, and to implement drawing page breaks on the screen. The other print handler method related to page breaks is `GetBreakCoord`. This method returns the location of an arbitrary page break, in view coordinates. If the strip size is fixed then this is easily computed by multiplying the strip size by page break number. If the strip size is variable, then the break location is computed by calculating each and every page break up to the one `GetBreakCoord` is interested in. To cut down on these computations, `TStdPrintHandler` caches the last computed page break computed.

Screen Feedback

In many applications it is desirable to show on the screen where page breaks occur in a view. For this purpose, print handlers support the notion of screen feedback. Screen feedback can be used to draw any printing related info, such as page breaks, page numbers, and so on.

The view initiates screen feedback by calling its `DoDrawPrintFeedBack` method from `DrawContents`. Unless overridden, it in turn calls the print handler's `DrawPrintFeedback`. If `TStdPrintHandler.fShowBreaks` is true, page breaks will be drawn on the screen by calling the view's `DoDrawPageBreak` method. The default `DoDrawPageBreak` method turns around and calls the print handler's `DrawPageBreak` method. For `TStdPrintHandler`, this simply draws a two-pixel wide gray line along the page break. As a debugging aid, if `gDebugPrinting` is true, then `TStdPrintHandler.DrawPrintFeedback` will also draw page numbers at the intersections of the page breaks.

`TStdPrintHandler.InvalPageFeedback` is used to show or hide page feedback. It is called in response to the Show/Hide Page Breaks command, or when the page breaks change, and simply invalidates the view.

How Printing Is Actually Done

Familiarity with the Printing Manager as described in Inside Macintosh, Volume IV, is helpful for reading this section.

Printing is done by the print handler method `Print`. It determines how many pages are to be printed, and then enters a loop in which it calls `TStdPrintHandler.OneSubJob` as many times as is necessary to completely print the view. For nonspooling print jobs the entire job will be completed with one call to `OneSubJob`. For spooled print jobs, however, it may be necessary to divide up the job into pieces, depending on the available disk space. `OneSubJob` calls `PrOpenDoc`, prints each page in the subjob by calling `TStdPrintHandler.PrintPage`, and calls `PrCloseDoc`.

PrintPage is responsible for printing one page of a view. It performs the following steps:

1. Calls TStdPrintHandler.SetPage to setup state information for the page being printed.
2. Calls PrOpenPage.
3. Calls TStdPrintHandler.FocusOnInterior to setup up focusing on the "interior" of the page and on the part of the view to be printed.
4. Calls TStdPrintHandler.DrawInterior, which in turn calls the view's DrawContents method. This will call the view's Draw method, passing an area rectangle that is the part of the view to be printed on the page, in QuickDraw coordinates.
5. Calls TStdPrintHandler.FocusOnBorder, in anticipation of drawing page adornments that may be outside the page interior.
6. Calls TStdPrintHandler.AdornPage to draw any page adornments such as page numbers, and so on.
7. Finally, PrClosePage is called.

Page Setup

TStdPrintHandler implements the Page Setup menu command and the page setup dialog. The changes from the page setup dialog are undoable. This is done by creating a TPrintStyleChangeCommand command object if the user clicks OK in the page setup dialog.

Menu Commands

Being subclasses of TEvtHandler, print handlers are capable of handling menu commands. TStdPrintHandler handles the following commands in its DoMenuCommand method:

cPrint	Calls TStdPrintHandler.CheckPrinter to ensure that the print handler is in sync with the selected printer, and then calls TStdPrintHandler.PoseJobDialog to display the job dialog. If the user clicks OK, Print is called to print the view.
cPrintOne	Calls TStdPrintHandler.CheckPrinter to ensure that the print handler is in sync with the selected printer, and then calls TStdPrintHandler.SetupPrintOne. SetupPrintOne simply validates the print record and sets the first page to 1 and the last page to 9999. Then Print is called to print the view.
cPageSetup	Calls TStdPrintHandler.PosePageSetupDialog and returns its result. PosePageSetupDialog will display the page setup dialog, and if the user clicks OK, a command object of type TPrintStyleChangeCommand is returned, so that the page setup is undoable.
cShowBreaks	Toggles the value of the fShowBreaks field of the print handler and calls TStdPrintHandler.InvalPageFeedback to either hide or show the page feedback.

Note that print handlers are not installed in the target chain, so their DoMenuCommand must be explicitly called by a view or document. This is done in TView.DoMenuCommand and TDocument.DoMenuCommand. The same applies to the print handler's DoSetupMenus.

View Printing Methods

The view printing methods fall into two categories. Some of them are called from the print handler, and allow customization of printing without overridding the print handler. The others are intended to be called from the view or from your code.

Printing Methods Called from the Print Handler

```
PROCEDURE TView.AttachPrintHandler (itsPrinthandler: TPrintHandler);
```

This method simply sets the `fPrintHandler` field of the the view to `itsPrintHandler`. It is called from `TPrintHandler.IPrintHandler`.

```
FUNCTION  TView.DoBreakFollowing (vhs: VHSelect;  prevBreak: VCoordinate;
    VAR automatic: BOOLEAN): VCoordinate;
```

This method returns the view coordinate of the page break following `prevBreak` in the direction `vhs`. `automatic` is set to true if this is an "automatic" page break, and to false if it's a "manual" page break. The default behavior is to call `fPrintHandler.BreakFollowing` to compute the coordinate, with `automatic` is set to true. This method is called from the print handler whenever it needs to compute page breaks. Either this method, or `fPrintHandler.BreakFollowing` must be overridden for views that have varying page strip sizes.

```
PROCEDURE TView.DoCalcPageStrips (VAR pageStrips: Point);
```

This method determines the number of page strips for the view and returns it in `pageStrips`. The default behavior is to call `fPrintHandler.CalcPageStrips` and return its result. This method is called from the print handler whenever it needs to recompute the number of page strips. You may wish to override this method for variable strip sizes in order to implement a more efficient way of determining the number of page strips. (If you don't, and the strip sizes vary, the print handler will compute every single page break to determine the total number.)

```
PROCEDURE TView.DoCalcViewPerPage (VAR viewPerPage: VPoint);
```

This method determines the default size of a page strip. The default behavior is to call `fPrintHandler.CalcViewPerPage`, which returns the size of the page interior. This method is called from the print handler when it is created and when the printer changes.

```
PROCEDURE TView.DoDrawPageBreak (vhs: VHSelect; whichBreak: INTEGER;
    loc: VCoordinate; automatic: BOOLEAN);
```

This method is called by the view's print handler to draw a page break on the screen. The default behavior is to call `fPrintHandler.DrawPageBreak`, which draws a two-pixel wide gray line along the page break. You can override this method if you wish to draw page breaks differently, or if you want to visually differentiate between automatic and manual page breaks.

```
PROCEDURE TView.DoPagination;
```

This method is called when the view's pagination changes. Specifically, this is called when the view's size changes, or when the printer changes. Furthermore, you may call this method when any change is made that affects pagination of the view. The default behavior is to call `fPrintHandler.RedoPageBreaks`. For views with varying page strip sizes, this can be costly in that the print handler will have to calculate all of the page breaks (unless you override `TStdPrintHandler` and/or the appropriate view methods.

```
PROCEDURE TView.DoPrinterChanged;
```

This method is called by the print handler when it detects that the printer has changed. The default behavior is to call `fPrintHandler.PrinterChanged`. You may override this if the view needs to repond to a new printer in a nonstandard way.

```
PROCEDURE TView.DoSetPageOffset (coord: VPoint);
```

This method is called by the print handler when it is preparing to print a page. The default version calls `fPrintHandler.SetPageOffset`.

```
PROCEDURE TView.GetPrintExtent (VAR printExtent: VRect);
```

This method returns the part of the view that is printable. The default behavior is to return the view's extent. You can override this method if only part of the view's extent is to be printed.

```
PROCEDURE TView.PageInteriorChanged (newInterior: Rect);
```

This method is called by the print handler when the page interior changes, giving the view a chance to react to the change. The default behavior does nothing.

```
PROCEDURE TView.DoCheckPrinter;
```

This method calls `fPrintHandler.CheckPrinter` to see if the printer has changed.

```
PROCEDURE TView.DoDrawPrintFeedback (area: Rect);
```

This method is called from the view's `DrawContents` method to draw printing feedback. The default behavior is to call `fPrintHandler.DrawPrintFeedback`.

The TStdPrintHandler Class

Fields Inherited From TPrintHandler

`fView`	The view whose image this print handler prints.
`fPageAreas`	A record defining the dimensions of the printed page. It contains the following fields:
• `theInk`	A rectangle defining the printable part of the page. Its top-left corner is always (0, 0). Its bottom-right corner is the maximum page height and width attainable on the given printer.
• `thePaper`	A rectangle defining the entire physical page, in the coordinate system of theInk. Since the physical page is typically larger than the printable part of the page, the top-left corner of thePaper is negative and the bottom-right coordinates are greater than those of `theInk`.
• `theInterior`	A rectangle defining the part of the page in which the print handler draws its view, in the coordinate system of `theInk`. `theInterior` is never larger

than `theInk`. It may be smaller than `theInk` if the application wishes to inset printing from the edges of the paper.

- `theMargins` — A rectangle defining the top, left, bottom, and right paper margins. The margins define the distance from the edge of the physical page to the interior of the page. The margins can never be smaller than the difference between `theInk` and `thePaper`. `theInterior` is computed by subtracting `theMargins` from `thePaper`.

`fDeviceRes` — The resolution of the printer, in dots per inch.

`fEffectiveDeviceRes` — The true effective resolution of the printer. It is larger than `fDeviceRes` if reduction is in effect, smaller if enlargement is in effect, or the same if neither is in effect.

`fViewPerPage` — The default page strip size.

`fFocusedPage` — The page number of the currently focused page during printing.

TStdPrintHandler Fields

`fDocument` — The document with which this print handler is associated.

`fPrintExtent` — The part of the view's extent that is to be printed by the print handler. It is determined by calling the view's `GetPrintExtent` method.

`fFixedSizePages` — Indicates whether the page strips are of fixed size horizontally and vertically.

`fHPrint` — A handle to the Printing Manager print record.

`fPageStrips` — The number of horizontal and vertical page strips.

`fStartPage` — The page number of the first page.

`fPrinterDev` — The device number of the printer, as returned by the Printing Manager.

`fLastCheckedPrinter` — The tick count at the last time it was known that the print handler was in sync with the choosen printer.

`fLastPrinterName` — The name of the printer driver at the time of `fLastCheckedPrinter`.

`fPageDirection` — Indicates whether page numbering is horizontal or vertical. Vertical means that page 2 is under page 1, and so on. Horizontal means that page 2 is to the right of page 1, and so on.

`fShowBreaks` — Indicates whether page breaks should be shown when the view is drawn on the screen.

`fFinderSetup` — Indicates whether to pose the Page Setup dialog when Finder printing.

`fFinderJobDialog` — Indicates whether to pose the job dialog when Finder printing.

`fSquareDots` — Applies to the ImageWriter only. If true, use 72-by-72 dot resolution. If false, use 80-by-72 dot resolution ("tall adjusted").

`fMinimalMargins` — If true, page margins are maintained such that exactly the complete printable area

	of the page is used (`fPageAreas.theInk` and `theInterior` are the same).
`fLastStrip`	Caches the last page strip whose page break was computed.
`fLastBreak`	Caches the coordinates of the last page break computed.
`fViewedRect`	Caches the part of the view that is visible in the current page.

Page Strip Methods

```
PROCEDURE TStdPrintHandler.CalcViewPerPage (VAR amtPerPage: VPoint); OVERRIDE;
```

This method implements the standard technique for determining the default page strip size. `amtPerPage` is set to the size of the paper (`fPrintAreas.thePaper`) minus the margins (`fPrintArea.theMargins`).

```
PROCEDURE TStdPrintHandler.CalcPageStrips (VAR pageStrips: Point); OVERRIDE;
```

This method implements the standard technique for determining the number of horizontal and vertical page strips in `pageStrips`. For each dimension, if it has fixed size pages then the number of page strips is computed by dividing the strip size by the print extent. Otherwise, the number of page strips is computed by calculating every page break and adding them up. This method is called from `TView.DoCalcPageStrips`. To implement a different technique, you can override `TView.DoCalcPageStrips` or this method.

```
FUNCTION TStdPrintHandler.BreakFollowing (vhs: VHSelect; prevBreak:
    VCoordinate; VAR automatic: BOOLEAN): VCoordinate; OVERRIDE;
```

This method returns the coordinate of the page break following `prevBreak`. Simply adds the page strip size to `prevBreak` and sets automatic to true. `vhs` indicates whether the page break is horizontal (that is, it separates rows of page strips) or vertical (that is, it separates columns of page strips).

```
PROCEDURE TStdPrintHandler.GetBreakCoord (vhs: VHSelect; whichBreak: INTEGER;
    VAR loc: VCoordinate);
```

This method returns the view coordinate for the given page break. `vhs` indicates whether the page break is horizontal (that is, it separates rows of page strips) or vertical (that is, it separates columns of page strips). whichBreak is the page break number, where the first page break is 1. The coordinate of the page break is returned in loc. After computing the page break location, `GetBreakCoord` caches the page break in `fLastStrip.vh[vhs]` and `fLastBreak.vh[vhs]`. This method is used to determine the part of the view printed on each page during printing.

To optimize printing performance, this method attempts to use the best technique it can to determine the page break coordinate. The algorithm used to determine loc depends on a number of factors. If the page strip size in the `vhs` direction is fixed, then `loc` is easily determined by multiply the page strip size by `whichBreak`. Otherwise, if the requested page break is the same as the last one cached (`whichBreak` equals `fLastStrip.vh[vhs]`), then `loc` is set to `fLastBreak.vh[vhs]`. Finally, if this second condition is not met, page breaks are computed one at a time, starting with the last strip cached (or 1 if the last strip is beyond `whichBreak`), by calling `fView.DoBreakFollowing` until the break coordinate has been determined for `whichBreak`.

```
PROCEDURE TStdPrintHandler.EachBreak (vhs: VHSelect; includeLast: BOOLEAN;
    FUNCTION DoToBreak (loc: VCoordinate; automatic: BOOLEAN): BOOLEAN);
```

This method iterates through the page breaks in the `vhs` direction, calling `DoToBreak` for each page break. It terminates when `DoToBreak` returns true or when `DoToBreak` has been called for the last page break (or for the next-to-the-last if `includeLast` is false). The page breaks are determined by calling `fView.DoBreakFollowing`. This method is used by `CalcPageStrips` if the page strips are of varying sizes, and by `DrawPrintFeedback` to cause page breaks to be displayed.

```
FUNCTION  TStdPrintHandler.PageToStrip (pageNumber: INTEGER); Point;
```

This method returns the horizontal and vertical page strip for the given page number.

```
FUNCTION  TStdPrintHandler.StripToPage (hStrip, vStrip: INTEGER): INTEGER;
```

This method returns the page number for the given horizontal and vertical page strip.

```
FUNCTION TStdPrintHandler.PointToPageStrip (pointInView: VPoint): Point;
```

This method returns the page strip in which the given view point is located.

Utility Methods

```
PROCEDURE TStdPrintHandler.GetDocName (VAR docName: Str255);
```

This method returns a name that is used in the printing dialogs and error messages. If `fDocument` is `NIL`, then `docName` is set to an empty string. Otherwise, if `fDocument` has no windows then `fDocument.fTitle` is returned. If this second condition is not met, the name of the document's first window in its window list is returned.

```
PROCEDURE TStdPrintHandler.DoInMacPrint (PROCEDURE WhatToDo);
```

This method sandwiches a call to `WhatToDo` with `PrOpen` and `PrClose`. Used to perform actions that need to be done while the Print Manager is open.

```
FUNCTION  TStdPrintHandler.MaxPageNumber: INTEGER; OVERRIDE;
```

This method returns the highest page number that would be printed if the user requested that all pages be printed.

Print Formatting and Setup Methods

```
PROCEDURE TStdPrintHandler.CheckPrinter; OVERRIDE;
```

This method checks to see if the user has changed the printer specifications (by choosing another printer, for example). If the specifications have changed,then the view's `DoPrinterChanged` method is called.

```
PROCEDURE TStdPrintHandler.PrinterChanged; OVERRIDE;
```

This method is called from `TView.DoPrinterChanged`. It calls the view's `DoPagination` method.

```
PROCEDURE TStdPrinthandler.RedoPageBreaks; OVERRIDE;
```

This method is called from `TView.DoPagination`. Calls `SetPrintExtent` to get the view's print extent, calls `SetMargins`, resets the page interior and forces page breaks to be recomputed.

```
PROCEDURE TStdPrintHandler.SetPrintExtent;
```

This method sets the print extent (the part of the view's extent that is to be printed) by calling the view's `GetPrintExtent` method.

```
PROCEDURE TStdPrintHandler.SetMargins;
```

This method sets the print handler's margins by calling `InstallMargins`.

```
PROCEDURE TStdPrintHandler.InstallMargins (newMargins: Rect;
    areMinimalMargins: BOOLEAN);
```

This method installs the margins by setting `fPageAreas.theMargins` and `fPageAreas.theInterior`.

```
PROCEDURE TStdPrintHandler.ValidatePrintRecord (VAR didChange: BOOLEAN);
```

This method validates the Print Manager print record by calling the Print Manager routine `PrValidate` and returning its result in the `didChange` parameter.

```
PROCEDURE TStdPrintHandler.Reset;
```

This method resets the Print Manager print record associated with the print handler.

```
PROCEDURE TStdPrintHandler.SetDefaultPrintInfo;
```

This method creates a Print Manager print record and initializes it to the default values by calling `Reset`.

```
FUNCTION TStdPrintHandler.SetupForFinder: BOOLEAN; OVERRIDE;
```

This method sets up the print handler for Finder printing. `PosePageSetupDialog` is called if `fFinderPageSetup` is true. `PoseJobDialog` is called if `fFinderJobDialog` is true or if this is the first document being printed. Otherwise, the Print Manager routine `PrJobMerge` is used to merge the job information from the previous job. `SetupForFinder` returns true if the application should continue printing.

```
FUNCTION TStdPrinthandler.SetupPrintOne: BOOLEAN;
```

This method sets up the print handler for "Print One" printing, which does not use a job dialog. Instead, it simply validates the print handler's print record and sets the first and last pages to zero and 9999.

Actual Printing Methods

```
FUNCTION TStdPrintHandler.Print (itsCmdNumber: CmdNumber;
    VAR proceed: BOOLEAN): TCommand; OVERRIDE;
```

This method carries out printing for the print handler. `Print` determines how many pages are to be printed, and then enters a loop in which it calls the method `TStdPrintHandler.OneSubJob` as many times as is necessary to completely print the view. For nonspooling print jobs, the entire job will be completed with one call to `OneSubJob`. For spooled print jobs, however, it may be necessary to divide up the job into pieces, depending on the available disk space.

```
FUNCTION TStdPrintHandler.OneSubJob (subjobFirstPage, subjobLastPage: INTEGER;
    justSpool: BOOLEAN; partialJob: BOOLEAN; VAR ranOutOfSpace: BOOLEAN;
    VAR lastPageTried: INTEGER; VAR proceed: BOOLEAN): TCommand;
```

This method prints one *subjob*, where the subjob consists of the pages `subjobFirstPage` to `subjobLastPage`. (To print an entire job, it may be necessary to break it up into subjobs if spool printing is in effect and there isn't enough disk space to spool the entire job.) The subjob is printed by calling the Print Manager routine `PrOpenDoc`, printing the required pages, and calling `PrCloseDoc`. For spooled jobs, each page is printed as many times as is required by the job. For nonspool jobs, each page is printed once and the printer itself reproduces the copies if needed. `ranOutOfSpace` returns true if `OneSubJob` has run out of disk space while spooling pages. In that case, `lastPageTried` is the page in which `OneSubJob` ran out of space. `proceed` is false if the user cancels the job, or an error occurs causing the job to be aborted.

```
PROCEDURE TStdPrintHandler.PrintPage (aPageNumber: INTEGER);
```

This method prints one page. It performs the following steps:

1. Calls `TStdPrintHandler.SetPage` to set up state information for the page being printed.
2. Calls `PrOpenPage`.
3. Calls `TStdPrintHandler.FocusOnInterior` to set up up focusing on the interior of the page and on the part of the view to be printed.
4. Calls `TStdPrintHandler.DrawInterior`, which in turn calls the view's `DrawContents` method. This will call the view's `Draw` method, passing an area rectangle that is the part of the view to be printed on the page, in QuickDraw coordinates.
5. Calls `TStdPrintHandler.FocusOnBorder`, in anticipation of drawing page adornments that lie outside the page interior.
6. Calls `TStdPrintHandler.AdornPage` to draw any page adornments such as page numbers, and so on.
7. Calls `PrClosePage`.

```
PROCEDURE TStdPrintHandler.SetPage (aPageNumber: INTEGER);
```

This method sets up the print handler to print the given page. The part of the view to be printed is determined by calling `GetBreakCoord` four times to get the horizontal and vertical page breaks that define the page. This part of the view is saved in `fViewedRect`. `SetPageInterior` is called to set `fPrintAreas.theInterior` for the given page. `fView.DoSetPageOffset` is called to set the global variable `gPageOffset`. The page being printed is saved in `fFocusedPage`.

```
PROCEDURE TStdPrintHandler.SetPageInterior (pageNumber: INTEGER): OVERRIDE;
```

This method is responsible for installing the correct values into `fPageAreas.theInterior`, representing the interior of the page in view coordinates.

```
PROCEDURE TStdPrintHandler.LocatePageInterior (pageNumber: INTEGER;
    VAR loc: Point); OVERRIDE;
```

This method returns in loc the top-left corner of the page interior for the given page. Unless overridden, it returns the sum of the top-lefts of `fPageAreas.thePaper` and `fPageAreas.theMargin`.

```
PROCEDURE TStdPrintHandler.SetPageOffset (coord: VPoint): OVERRIDE;
```

This method sets the global variable `gPageOffset` to the given coordinate.

```
PROCEDURE TStdPrintHandler.FocusOnInterior; OVERRIDE;
```

This method focuses on the page interior of page number `fPageFocused`. This is analogous to focusing views through windows, except this method causes views to be focused onto a page. Like `Focus`, this method sets the printing port origin (to `gLongOffset` for large views) and the clipping region.

```
PROCEDURE TStdPrintHandler.FocusOnBorder;
```

This method focuses on the entire page whose number is `fPageFocused`, so that page adornments can be drawn outside the page's interior.

```
PROCEDURE TStdPrintHandler.DrawPageInterior;
```

This method draws the page interior by calling the view's `DrawContents` method. The view will be focused onto the page by the `FocusOnInterior` method.

```
PROCEDURE TStdPrintHandler.AdornPage;
```

This method is used to draw page adornments that are not drawn by the view itself, such as page numbers. If `gDebugPrinting` is true, then `AdornPage` prints page numbers and boxes the interior and border of the page.

Printing-Related Dialogs

```
FUNCTION TStdPrintHandler.PoseJobDialog: BOOLEAN;
```

This method displays the printer's job dialog by calling `PrJobDialog` and returning its result. After `PrJobDialog` is completed, `PoseJobDialog` causes all windows to be updated after the dialog is removed from the screen. (This is because control will not return to the main event loop until after printing is completed, so the windows are updated here to avoid "white space" while printing takes place.)

```
FUNCTION TStdPrintHandler.PosePageSetupDialog (VAR proceed: BOOLEAN;
    isUndoable: BOOLEAN): TCommand;
```

This method displays the printer's page setup dialog by calling `PrStlDialog`. `proceed` is set to the result of `PrStlDialog`. If `isUndoable` is true, then the a `TPrintStyleChangeCommand` object is created so that changes made in the dialog are undoable, and the command object is returned as this method's result. Otherwise, `gNoChanges` is returned.

```
PROCEDURE TStdPrintHandler.PosePrintDialog;
```

This method displays the print dialog and leaves it on the screen. The standard print dialogs are in the resource file Printing.r. Their resource IDs are `phFinderPrintDialog` and `phSpoolPrintDialog`. The `phFinderPrintDialog` dialog is used when printing from the Finder. This dialog has a Cancel button for canceling the current job, and a Cancel All button for cancelling the current and subsequent jobs. The `phSpoolPrintDialog` is used in all other cases, and it has only a Cancel button.

The print dialog is removed from the screen by the `BanishPrintDialog` method.

```
PROCEDURE TStdPrintHandler.ShowDocBeingPrinted (entering: BOOLEAN);
```

This method creates (entering is true) or disposes of (entering is false) a dialog used by the Print Manager to get the title of the document being printed. The dialog's ID is `phWhichDoc`, and its title is set to the title returned by `GetDocName`. The dialog is never made visible. The reason for doing this is that the Print Manager gets the document name from the front window during `PrValidate`, and when the user is printing from the Finder there are usually no open windows. The dialog is disposed of by the `BanishPrintDialog` method.

```
PROCEDURE TStdPrintHandler.BanishPrintDialog;
```

This method disposes of the print dialog displayed from `ShowDocBeingPrinted` or `PosePrintdialog`, both of which set `gPrintDialog` to the dialog they create.

Screen Feedback Methods

```
PROCEDURE TStdPrintHandler.DrawPrintFeedback (area: Rect); OVERRIDE;
```

This method implements the standard print feedback while its view is displayed on the screen. It is called from `TView.DoDrawPrintFeedback`. `area` is the part of the view that is being redrawn, in QuickDraw coordinates. If `fShowBreaks` or `gDebugPrinting` is true, then this method will call `fView.DrawPageBreak` for every page break visible in area.

```
PROCEDURE TStdPrintHandler.DrawPageBreak (vhs: VHSelect; whichBreak: INTEGER;
    loc: VCoordinate; automatic: BOOLEAN); OVERRIDE;
```

This method implements the standard way of drawing a page break in the print handler's view while it is displayed on the screen. It is called from `TView.DoDrawPageBreak`. It draws a two-pixel wide gray line at each page break, and if `gDebugPrinting` is true, then it draws the page numbers at the intersections of the vertical and horizontal page breaks.

```
PROCEDURE TStdPrintHandler.InvalPageFeedback;
```

This method invalidates page feedback in the print handler's view by calling the view's `ForceRedraw` method. It is called when the page breaks change, or when the state of `fShowBreaks` changes.

Menu Methods

```
FUNCTION TStdPrintHandler.DoMenuCommand (aCmdNumber: CmdNumber): TCommand;
    OVERRIDE;
```

This method handles the commands associated with the print handler. Since print handlers are not normally a part of the target chain, this method is explicitly called from the view (or document) associated with the print handler. The commands handled are

`cPrint`	Prints the print handler's view by calling `CheckPrinter` to verify the printer, posing the job dialog, and if the user clicks OK, calling `Print`.
`cPrintOne`	Prints the print handler's view by calling `CheckPrinter` to verify the printer, and then calling `Print`.
`cPageSetup`	Poses the page setup dialog.
`cShowBreaks`	Toggles the choice of whether or not the print handler should provide page break feedback on the screen.

```
PROCEDURE TStdPrintHandler.DoSetupMenus; OVERRIDE;
```

This method sets up the commands associated with the print handler. Since print handlers are not normally a part of the target chain, this method is explicitly called from the view (or document) associated with the print handler.

Miscellaneous Methods

```
PROCEDURE TStdPrintHandler.ChooseSpoolFile (VAR spoolFileName: Str255; VAR
    spoolVRefNum: INTEGER; VAR pagesPerSubjob: INTEGER);
```

This method returns the filename and volume of the print spool file, and the number of pages to be spooled before printing the spooled file. This is only relevant for spooled printing. Unless overridden, this method returns an empty string in `spoolFileName` and zero in `spoolVRefNum` (indicating that the print driver should use its standard file name) and returns `MAXINT` for `pagesPerSubjob`.

```
PROCEDURE TStdPrintHandler.Terminate; OVERRIDE;
```

This method is called by MacApp when the application is being terminated, for the print handler installed in `gPrintHandler`. It simply enables the Chooser by calling `EmpowerChooser(TRUE)`.

Creation/Destruction Methods

```
PROCEDURE TStdPrintHandler.IStdPrintHandler (itsDocument: TDocument;
    itsView: TView; itsSquareDots, itsHFixedSize, itsVFixedSize: BOOLEAN);
```

Initializes a `TStdPrintHandler` object. Calls `IPrintHandler`, and then calls `itsView.AttachPrintHandler` to notify the view that it is associated with a print handler. Initializes the print handler fields as follows:

`fDocument`	Set to `itsDocument`.
`fHPrint`	If the document shares its print handler, then `fHPrint` is set to the document's print record. Otherwise, a new print record is created and initialized by the `Reset` method.
`fStartPage`	Set to 1.
`fPageDirection`	Set to `vertical`.
`fShowBreaks`	Set to false.
`fFixedSizePages`	Set to `itsHFixedSize` and `itsVFixedSize`.
`fViewPerPage`	Set to (32, 32). Will be changed by calling the view's `DoCalcViewPerPage`.
`fFinderSetup`	Set to false.
`fFinderJobDialog`	Set to false.
`fSquareDots`	Set to `itsSquareDots`.
`fLastStrip`	Set to `(Maxint, Maxint)`.
`fLastBreak`	Set to (0, 0).
`fPageAreas.theMargins`	Set to `gStdPageMargins`.
`fMinimalMargins`	Set to true.