PrintTwist™

Version 2.1.2

Developers Reference

A 4D Component based Report Generator

By Tony Ringsmuth
of
Business Brothers, Inc.
Also known As
BBI

Revision 03/05/2008

Copyright 2002 to 2008 by Business Brothers, Inc.
All Rights Reserved

Please read licensing agreement before installing PrintTwist

Preface

PrintTwist is designed to be a companion to and a command level competitor for PrintList™ by Automated Solutions Group. PrintTwist was written from scratch in native 4D. Several printing commands that were added to 4th Dimension™ in version 6.8.1 made the creation of a 4D native based product such as PrintTwist possible.

In this document, PrintList™ may be referred to as PLP (PrintList™ Pro). PrintTwist may be referred to as PT.

PrintTwist is compatible with 4D versions 6.8.2 to 11.X.

Comparison of PrintTwist and PrintList™

While PrintTwist is designed to work as similarly as possible to PrintList™, there are some key differences that the developer needs to understand. The following is a discussion of the technical differences between PrintTwist and PrintList.

Technical differences between PrintList™ and PrintTwist	
PrintList™	PrintTwist™
PLP is a plug-in. It is installed in a Mac4DX or a Win4DX folder near the database template or 4D application. You need a platform specific version (Mac or PC).	PT is a 4D Component. It is installed directly into your 4D Template using 4D Insider, and appears as 4D Methods and forms. It automatically runs on all platforms supported by 4th Dimension.
PLP is used with the Print Selection command or the Print Record command. All PLP configuration commands are called from within the form method, or a sub-method of the form method.	PT uses 4D's Print Form command. PT Configuration takes place outside of the form which will be printed.
PLP uses a Pluggin area on a form to act as an object in which the report will print into.	There are two ways to designate a PT area: 1: you can create a rectangle object on a form 2: you can specify the boundaries of your PT area programmatically, without any actual object on a form.
PLP is always fast	Because PT may run in source code or compiled, the speed will vary greatly. Running in source, is NOT fast. If you need to print long reports running in source code, then PrintList™ should be your solution. While PT is accelerated when compiled, it still does not match the speed or PLP.
When printing Fields and records, you cannot mix fields and Arrays	PrintTwist does allow you to mix arrays and fields in the same report.`
PrintList has no method to pre-determine the length (page count) of a report before printing.	PrintTwist allows you to know how many pages long a report will be before printing.
	PrintTwist has some additional Calculations that it can perform for you when using break level processing, above the normal PrintList break calculation features

Maximum limit comparison	
PrintList™	*PrintTwist™*
32,000 rows	2 billion rows (in reality, limited by memory)
100 columns	255 columns

Design

While the nitty-gritty of configuring a PrintTwist area is nearly identical to PrintList™, there is a significant difference the primary calls to PrintTwist™

First, we must define some basic terms.

Host Forms

A host form is a form that contains one or more PT Areas, and usually some of your own custom objects. A host form acts as a host for your PrintTwist report. It will normally have a header area and/or a footer area. Your host form can look any way you want it to look. It may or may not have a PT Area drawn on the host form. You can configure the PT areas on a host form either programmatically, or by drawing rectangle objects to act as a PT Area.

You specify the form table, form name, and marker coordinates of the host form in the call to ***PT_SetHostForm***.
Example: ***PT_SetHostForm*** (->[HostFormTable];"MyPTHostForm";50;650;650;700)

You may use PrintTwist™ without a host form. If you do this, the PT Areas will begin print at the very top of the printed page, and extend to the lowest possible area of each page.

PrintTwist™ cannot read the Header, Detail, Break or Footer markets that are configured in a 4D Form. If you are using a hostform, you must pass in these coordinates as $3 through $6.

If you want to visually see the location of the PT Area on your form, then draw a rectangle object, and name the rectangle object. The name that you give the rectangle will be the name that you pass in to PT_GetAreaRef.

No objects to the right or the left of the PT Area will be printed. Only objects above or below the PT Area will be printed.

PT Areas

A PT Area is a target area into which a PrintTwist prints a report.

You may create a PT Area graphically on a host form, by drawing a rectangle, and naming the rectangle, or you may create a logical PT Area with the use of the ***PT_GetAreaRef*** method.
Examples:
vReportRef:=***PT_GetAreaRef***("MyPT_Area";{Opts};{Left;Top;Right;Bottom})

vReportRef:=***PT_GetAreaRef***("MyPT_Area";0;1;50;vMyPageWidth;vMyPageHeight-50)

See also the section on ***PT_getAreaRef***.

PT Object Forms

PT Object Forms are the forms that are installed into your database with the PrintTwist™ component. These forms contain all the lines and variables that PrintTwist™ uses to print your reports.

PrintTwist™ installs several forms which might appear to be redundant of eachother. Each of these forms will have a numeric suffix on the name. The reason for this is that when you are printing a small report, PT can work faster with a form containing fewer objects. As PT prints, depending on the number of columns your report has, PT will select the smallest possible PT Objects form that still has enough columns.

Each of the PT Objects forms must call the Project Method, ***PT_ObjectsFormMethod***.

Language

When programming with PrintTwist™, it is important to know that all PrintTwist™ report configuration is done outside of (and before) the form method of the printed report. Here is the basic order of using the PrintTwist™ Language:

1: Build your table of arrays. (this can be done any time prior to using ***PT_SetArraysNam*** or ***PT_SetArraysPtr or PT_AppendColumn***)

2: Get a PrintTwist Area Reference: (***PT_GetAreaRef***)

3: Pass your arrays to PrintTwist (***PT_SetArraysNam*** or ***PT_SetArraysPtr or PT_AppendColumn***)

4: Configure the report (All PT_ commands that take a PT Area reference)

5: Configure the HostForm (***PT_SetHostForm***: This can be done any time prior to ***PT_Print***)

6: optionally, you can call ***PT_GetPageCount*** to get a count of the number of pages that will print.
6: Print the report: (***PT_Print***: This is always the final step.)

Here is a basic example of a PrintTwist report method:

```
`My PT Example

`Load some arrays with data
ALL RECORDS([Invoice])
SELECTION TO ARRAY([Invoice]Total;aInvTotal;
          [Invoice]Invoice_Date;aInvDate; [Client]Name;aInvClientName)

`Create a PrintTwist area reference to the PT Object on [BB_MISC];"PT_Main1"
`We need this reference number to do all other configuration involving this
`   instance of a this PT_Area.
$iPTArea:=PT_GetAreaRef ("PT_Area1")

`Add the arrays into the report
PT_SetArraysNam ($iPTArea;1;3;”aInvClientName”;”aInvDate”;”aInvTotal”)

`Add the headers into the report
PT_SetHeaders ($iPTArea;1;3;"Client Name";"Invoice Date";"Total")

`Sort by Client Name and then Invoice Date
PT_SetSort ($iPTArea;1;2)   `

`Make a subtotal by client
PT_SetBrkText ($iPTArea;1;3;"\Sum")


`Specify a Host form.   Set a 50 pixel header, 600 pixel body,
`    no Break area, and a 50 pixel footer
PT_SetHostForm (->[BB_MISC];"PT_Main1";50;650;650;700)

C_longint(vTotalPageCount)
vTotalPageCount:=PT_GetPageCount     `Get a page count so that we can print
          “Page {x} of {n}” on each page

PT_Print    `Print the report
`End of Method
```

Commands

Commands that differ from PLP

Each PrintTwist command has been designed to operate as closely as possible to the corresponding PrintList™ command, with all of it's nuances. Only the PT commands that function differently will be documented here, and only the differences in the command; NOT the full content of the command. If you require full documentation of how the command works, you must obtain the PrintList™ Pro developers reference from ASG.

Each PrintTwist command if prefixed with a PT_. Each PrintTwist command that replaces a PrintList™ command is spelled identical to the PrintList™ command except for the PT_ Prefix.

PT_SetArraysNam, PT_SetHeaders, PT_SetWidths

PT_SetArraysNam (AreaRef;StartColumn;NumColumns(ignored);"Array1" {;"ArrayN"})
PT_SetHeaders (AreaRef; StartColumn; NumColumns(ignored);"Header1" {;"HeaderN"})
PT_SetWidths (AreaRef; StartColumn; NumColumns(ignored);Width1 {;WidthN})

Each of these three commands works similarly to its PLP counterpart, with the following exceptions:.

If you pass a Zero in the StartColumn parameter ($2) for any of these commands, it will be interpreted as a 1.

NumColumns: The parameter is ignored.   You should pass either a zero, or the actual number of columns.

As where these PrintList™ commands can only receive up to columns at one time, these corresponding PrintTwist™ commands can receive up to 100 columns at a time.

PT_SetFrame, PT_SetDividers, PT_SetBrkRowDiv

For each of these three commands, the LineWidth settings are used to turn on or off the printing of the specified line. Any number greater than .01 will cause the line to print. User a Zero setting to avoid the printing of the line.

The Pattern and Color settings of these commands are ignored by PrintTwist™. You may modify the color or pattern or line width of any divider in PrintTwist™ Objects forms.

PT_SetCellStyle, PT_SetCellColor

For each of these two commands, the CellArray ($6) parameter must be a pointer to an array rather than just an array. The array may be of type integer, longint or real. If a pointer to anything other than a two-dimensional array is passed, it will be treated as a two dimensional array with zero elements.

IMPORTANT: If you use ***PT_SetCellStyle*** or ***PT_SetCellColor*** PRIOR to ***PT_SetSort***, the styles and colors will NOT sort with the data. This is an important difference of behavior between PLP and PT. Use ***PT_SetSort*** before ***PT_SetCellStyle*** or ***PT_SetCellColor*** if you need to keep specific rows with specific cell styles and colors.

PT_SetHeaders

PT_SetHeaders(AreaRef;StartColumn;NumColumns(ignored);Header1;{HeaderN})
PT_SetHeaders(AreaRef;StartColumn;NumColumns(ignored);Headers)

PT_SetHeaders may be used exactly like ***PL_SetHeaders***. In addition to this, you can also pass $4 as multiple headers, delimited by semicolons.

Examples:

PT_SetHeaders (MyArea;1;3;”First Name”;”Last Name”;”Phone”)
Or
PT_SetHeaders (MyArea;1;3;”First Name;Last Name;Phone”)

PT_SetBrkText, PT_SetBkHText

PT_SetBrkText(AreaRef; BreakLev; ColNum; BrkTxt; NumColToOverflow; Alignment)

PT_SetBkHText (AreaRef; BreakLev; ColNum; BrkTxt; NumColToOverflow; Alignment)

The Methods ***PT_SetBrkText*** and ***PT_SetBkHText*** have additional functionality over and beyond their PrintList counterparts.

The PrintTwist parameter BrkTxt ($4) can accept two additional calculation text features above and beyond their PrintList counterparts:

"\FirstValue"
If you pass "\FirstValue" for a break text or break header text, the first value for this column for break level will be printed.

"\LastValue"
If you pass "\LastValue" for a break text or break header text, the last value for this column for break level will be printed.

This feature is very useful when you are sorting and breaking on one column, such as "Client Name", but also have associated columns, such as "Client Address" where in the break, you want to show the break value of the client name, but you also want to print the associated address.

PT_SetColOpts

PT_SetColOpts(AreaRef; HideLastColumns; HideDetailArea; **ColumnPadding**)

$4: Longint: Column Padding

Parameters $1, $2, $3 function exactly the same as their PrintList™ counterparts. The 4th parameter, ColumnPadding is a PrintTwist supplement to the command PT_SetColOpts.

If you create a report that uses the default auto-calculation of column widths, PrintTwist employs an automatic padding of 7 pixels of width between the calculated data for each column. This is because the calculated font widths in memory do not always match what comes out on paper. This 7 pixels is a fudge-factor safty padding. If you find that PrintTwist needs to give some additional width to it's auto calculated column, then pass a value to ColumnPadding of greater than 7.

Supplemental commands in PrintTwist™

PrintTwist™ has a number of additional commands above and beyond the corresponding PrintList™ commands. These commands are documented in this section.

PT_HostFormMethod

PT_HostFormMethod ({->FromPixel};{->ToPixel};{->CurrentPrintingPage}) ->PrintingAreaName

Parameter	Type		Description
$0	Text	->	PrintingAreaName: Returns a string that describes the area of your host form that is currently bring printed Values Include: “TestPrint”: When printing an initial test print: zero pixels will be printed. “Header”: The general page header: to be printed on each page of the report. “Header_1”: Any detail space between the bottom of the page header and the top of PrintArea1. “AfterArea_{AreaRefString_X}”: Any space between PrintAreaX and PrintAreaX+1. “Break_0”: Space after the end of the final print area, down to the top of the page footer. “Footer”: The footer area to be printed on each page
FromPixel	Pointer	->	(output) Vertical Start Pixel: Pass a pointer to a numeric data type if you want to know the vertical print starting pixel of the form
ToPixel	Pointer	->	(output) Vertical End Pixel: Pass a pointer to a numeric data type if you want to know the vertical print end pixel of the form
CurrentPrintingPage	Pointer to numeric data type	->	(output) Current Printing Page

PT_HostFormMethod is used to help PrintTwist™ detect the PT Area object coordinates within your host form. This method does not require any parameters. You must call the method ***PT_HostFormMethod*** from your HostForm form method.

If you are manually configuring the coordinates of the PT Area through ***PT_GetAreaRef***, then you *may* avoid the usage of ***PT_HostFormMethod***.

Whenever your host form is printed, ***PT_HostFormMethod*** is called. When the PrintTwist areas are being printed, ***PT_HostFormMethod*** is NOT called.

Example method that you may have in your Host form:

```
`Form Method: My_PT_GenericForm_Port

`FUNCTION:
`This is the form method of the host form that has one or more
`    PrintTwist areas on it.
`Your host form method should always call PT_HostFormMethod, so that
`    PrintTwist can manage portions of the form.

C_LONGINT(vPrintFromPixel;vPrintToPixel;vPrintingPageNumber;vTotalPageCount)
$PrintingArea:=PT_HostFormMethod (->vPrintFromPixel;->vPrintToPixel;->vPrintingPageNumber)

`You do not need any of the following code, unless you want to manage
`    what happens during the printing of the various areas of your form.
Case of
    : ($PrintingArea="TestPrint")
    `No actual printing happens here: the printing should be from pixel zero to zero.
    `PrintTwist is doing this to detect where the PrintTwist areas on the form are.


  : ($PrintingArea="Header")
    `The general page header: This will happen once at the top of each page.
    vHeaderPageNumbers:="Page "+String(vPrintingPageNumber)+" of "+String(vTotalPageCount)


  : ($PrintingArea="Header_1")
    `Any detail space between the bottom of the page header and the
    `        top of PrintArea1.


  : ($PrintingArea="AfterArea_@")
    `AfterArea_{AreaRefString_X}: Any space between PrintAreaX and PrintAreaX+1.


  : ($PrintingArea="Break_0")
    `Space after the end of the final print area, down to the top of
    `        the page footer.


  : ($PrintingArea="Footer")
    `Footer:        The footer area to be printed on the bottom of each page


End case
`
```

PT_ObjectsFormMethod

PT_ObjectsFormMethod

PT_ObjectsFormMethod is used by PrintTwist™ to control printing. This method comes pre-installed into all PrintTwist™ PT_Object forms. The developer does not need to do anything additional with this method.

PT_SetHostForm

PT_SetHostForm (->[FormTable];"FormName";Header;Detail; BreakZero;Footer)

Parameter	**Type**	**In/out**	**Description**
FormTable	Pointer	->	A pointer to the table that has your "Host Form" with the PrintTwist areas that you will print
FormName	Text	->	The name of the form
Header	Longint	->	This vertical coordinate specifies the end of the Header area of the page. Everything above this coordinate will be printed at the top of each page in the report.
Detail	Longint	->	(This vertical coordinate specifies the end of the Detail area of the page. Everything from and including the Header coordinate and up to but not including the Detail coordinate is considered part of the detail area. The PT area must be fully contained in the Detail area.
BreakZero	Longint	->	This vertical coordinate specifies the end of the Break-Zero area of the page. The BreakZero zone will be printed just once, after the printing of all PT Areas, at the end of the PrintTwist™ report, just under the lowest PT Area.
Footer	Longint	->	This vertical coordinate specifies the end of the Footer area of the page. Everything from and including the BreakZero coordinate and up to but not including the Footer coordinate is considered part of the Footer area. The Footer area will be printed at the bottom of each page on the PrintTwist™ report.

PT_SetHostForm is used to set the hostform Table, form name, and marker coordinates of the Host Form.

Each of the marker coordinates specifies the end of it's designated zone. The exact pixel of each marker is actually included with the printing of the next zone. Each coordinate needs to be greater than or equal to the prior zone. (Detail should be farther down the form then header, etc…)

PT_GetAreaRef

PT_GetAreaRef (“AreaName”;{Opts};{Left;Top;Right;Bottom})->Longint Area Ref

#	Parameter	Type	In/out	Description
$0	Result	Longint	->	Reference to the PT Area
$1	AreaName	Text	->	Name of a PT Area on the host form
$2	Opts	Longint	->	Multiple Bitwise Options
$3	Left	Longint	->	Optional: Left coordinate of PT Area
$4	Top	Longint	->	Optional: Top coordinate of PT Area
$5	Right	Longint	->	Optional: Right coordinate of PT Area
$6	Bottom	Longint	->	Optional: Bottom coordinate of PT Area

PT_GetAreaRef is used to create a logical reference and memory space for a PT Area. ***PT_GetAreaRef*** must be called prior to any command that requires an Area Reference. The Area Reference that is produced by ***PT_GetAreaRef*** will be the first parameter of all other commands that configure a PT Area.

PT_GetAreaRef can be used in two different ways: in PrintTwist native mode, or PrintList™ Wrapper mode. If you use ***PT_GetAreaRef*** in PrintList™ wrapper mode, all subsequent calls to the specified PT Area will act as a wrapper to the corresponding PrintList™ commands. In this case, you will be actually printing your report using PrintList™, with PrintTwist as a wrapper around the PrintList™ commands.

To use ***PT_GetAreaRef*** in native PrintTwist™ mode:

AreaName - Text: The name of the PT Area object rectangle in the host form. You can pass a null string here if you do not have a PT Area drawn on the form. In this case, you will need to pass the left, top, right and bottom coordinates.

Opts – Longint: This is a bitwise options parameter. Options include:

+1: Run in PrintList™ wrapper mode. Select this option to print your report with the PrintList™ engine, instead of the PrintTwist™ engine. If you run in this mode, then all PrintTwist configuration calls must be made from within the form method of the PrintList™ form that is being used, just as if you were calling the PrintList commands directly. If you pass +1 here, then you must pass the PrintList™ area reference as $3 (Left), and then Result will return the reference to the PrintList™ area.

If you do NOT pass +1 in Opts, then PrintTwist™ will run in native PrintTwist mode.

+2: Run in PrintList™ Delayed Execution wrapper mode,. Select this option to print your report with the PrintList™ engine, instead of the PrintTwist™ engine.

This option is similar to +1: Wrapper mode, except that Delayed Execution allows you to pass the commands to PrintTwist before the printing of the form is called.

When using ***Delayed Execution wrapper mode***, you must pass in the string name of the PrintList™ object variable name as the AreaName Parameter.
Example:
vPrintRef:=***PT_GetAreaRef***(“vPrintRef”;2)
Then, in your PrintList™ form method, you call
PT_ExecutePLCode(AreaReference) to apply the PrintTwist configuration that you have to the PrintList area.

Left, Top, Right, Bottom – Longint: These parameters are optional, but should be passed or not passed as a group. You should pass values here if:

- You do not have a PT Area drawn on the host form
- You want to over-ride or adjust the actual coordinates of the specified PT Area.

AreaRef – Longint: A numeric reference to the PT Area. This reference will then be passed to each PrintTwist™ command that configures the PT Area.

PT_SetArraysPtr

PT_SetArraysPtr(AreaRef;ColumnNum;NumCols;->Array1 or ->Field1 {;->ArrayN or ->FieldN})
PT_SetArraysPtr(AreaRef;ColumnNum;NumCols;->ArrayGroup)

PT_SetArraysPtr is used like ***PT_SetArraysNam***, except that it takes pointers rather than array names within quotes.

PT_SetArraysPtr will accept any mix of pointers to arrays and fields.

This command can be called in two different ways:
- Traditional: by passing multiple individual arrays in $4-$n
- ArrayGroup: by passing a pointer to an array of pointers in $4

#	**Parameter**	**Type**	**In/out**	**Description**
$1	AreaRef	Longint	->	PT Area Reference
$2	ColumnNum	Longint	->	The starting column number for this set of arrays. *(new in v 1.9.0)* Pass 0 (zero) to append these columns to any previously passed columns.
$3	NumCols	Longint	->	The number of columns being passed. Pass zero for NumCols to be auto-calculated
$4 - $N	ColumnArrays	Pointer	->	Pointers to Arrays or Fields. You may pass a pointer to an array of pointers to the desired columns: in this case do not pass any parameters beyond $4.

Example:
PT_SetArraysPtr(AreaRef;1;0;->aFirstName;->aLastName;->aPhone)

Or
Array pointer(aGroup;3)
aGroup{1}:=->aFirstName
aGroup{2}:=->aLastName
aGroup{3}:=->aPhone
PT_SetArraysPtr(AreaRef;1;0;-> aGroup)

Passing fields to ***PT_SetArraysPtr***
When passing fields to ***PT_SetArraysPtr***, you can pass fields from related tables. The primary table that will be used for the fields is the table for the first field passed.

If the first field that you need to pass is not the primary table, then you must use ***PT_SetFile*** (AreaRef;TableNumber) to specify the primary table.

Technical Note: When printing fields, internally, PrintTwist gathers the field information into temporary process arrays via selection to array command. If you sort the list using PT_SetSort, it is these temporary arrays that get sorted, not your records.

Example:
PT_SetArraysPtr(AreaRef;1;0;->[invoice]balance;->[Client]Name;->aInvoiceAge)

PT_SetColumn

(New in v 1.9.0)

PT_SetColumn(AreaRef;ColNum;->Array;{HeaderText};{ColWidth};{Format};{ColumnJustify};{HeaderJustify})

PT_ SetColumn allows you to specify for one column at a time, the Array (or field), Header Text, Width, and Format. This command is very similar to ***PT_AppendColumn***, except for that with ***PT_SetColumn*** you also specify the column number.

Parameters 4 and above are optional: You may skip these parameters altogether, or pass null values for any of them.

#	Parameter	Type	In/out	Description
$1	AreaRef	Longint	->	PT Area Reference
$2	ColNumber	Longint	->	The column number If you pass zero here, then the column is appended as the new rightmost column of the report, exactly like ***PT_AppendColumn***
$3	Array	Pointer	->	The array or field for the column
$4	HeaderText	Text		Text to put in the header of the column
$5	Width	Longint	->	Pixel Width of the column. Pass zero
$6	Format	String	->	The format for the column (same as PT_SetFormat)

$7	Miscellaneous Parameters	Longint	->	ColJust;{ColJustNum}
				ColFont;{FontName}
				ColFontSize;{FontSize}
				ColFontStyle;{StyleNum}
				HdrJust;{HdrJustNum}
				HdrFont;{FontName}
				HdrFontSize;{FontSize}
				HdrFontStyle;{StyleNum}
				BrkText;{LevelNum};{Text}
				BrkOverflowCols;{LevelNum};
				BrkJust;{LevelNum};{ColJustNum}
				BrkFont;{LevelNum};{FontName}
				BrkFontSize;{LevelNum};{FontSize}
				BrkFontStyle;{LevelNum};{StyleNum}
				BkHText;{LevelNum};
				BkHOverflowCols;{LevelNum};
				BkHJust;{LevelNum};{ColJustNum}
				BkHFont;{LevelNum};{FontName}
				BkHFontSize;{LevelNum};{FontSize}
				BkHFontStyle;{LevelNum};{StyleNum}

Example:
eReport:=***PT_GetAreaRef*** ("ReportArea")

PT_SetColumn (eReport;1;->aFirstName;"First Name";40)

PT_ SetColumn (eReport;2;->aFirstLastName;"Last Name";40)
PT_ SetColumn (eReport;3;->aSalary;"Salary";40;"$###,##0.00";”ColJust;2”;
”HdrJust;2”;”BrkText;0;\SUM”

PT_AppendColumn

(New in v 1.9.0)
PT_AppendColumn(AreaRef;->Array;{HeaderText};{ColWidth};{Format}
;{MiscPrms1-N})

PT_ AppendColumn allows you to specify for one column at a time, the Array (or field), Header Text, Width, and Format. The passed column is always appended to the list of columns, so you do NOT need to pass a column number. This command is identical to PT_SetColumn, except that you do not pass a column number.

Parameters 3 and above are optional: You may skip these parameters altogether, or pass null values for any of them.

#	Parameter	Type	In/out	Description
$1	AreaRef	Longint	->	PT Area Reference
$2	Array	Pointer	->	The array or field for the column
$3	HeaderText	Text		Text to put in the header of the column
$4	Width	Longint	->	Pixel Width of the column. Pass zero
$5	Format	String	->	The format for the column (same as PT_SetFormat)

$6	Miscellaneous Parameters	Text	->	ColJust;{ColJustNum}
				ColFont;{FontName}
				ColFontSize;{FontSize}
				ColFontStyle;{StyleNum}
				
				HdrJust;{HdrJustNum}
				HdrFont;{FontName}
				HdrFontSize;{FontSize}
				HdrFontStyle;{StyleNum}
				
				BrkText;{LevelNum};{Text}
				BrkOverflowCols;{LevelNum};
				BrkJust;{LevelNum};{ColJustNum}
				BrkFont;{LevelNum};{FontName}
				BrkFontSize;{LevelNum};{FontSize}
				BrkFontStyle;{LevelNum};{StyleNum}
				
				BkHText;{LevelNum};
				BkHOverflowCols;{LevelNum};
				BkHJust;{LevelNum};{ColJustNum}
				BkHFont;{LevelNum};{FontName}
				BkHFontSize;{LevelNum};{FontSize}
				BkHFontStyle;{LevelNum};{StyleNum}

Example:

```
eReport:=PT_GetAreaRef ("ReportArea")

PT_AppendColumn (eReport;->aFirstName;"First Name";40)
PT_AppendColumn (eReport;->aFirstLastName;"Last Name";40)
PT_AppendColumn (eReport;->aSalary;"Salary";40;"$###,##0.00";
”ColJust;0”;”BrkText;0;\Sum”;”BrkFont;0;Geneva”;”BrkFontSize;0;12”;
”BrkFontStyle;0;1”)
```

PT_GetColNum

PT_GetColNum (AreaRef (or zero);->ColumnArray}) -> Column Number

#	Parameter	Type		Description
$0	*Column* Number	Longint	->	The column number for the specified *ColumnArray*
$1	*AreaRef*	Longint	->	Reference to a PrintTwist area to search. You only need to pass this parameter if there are multiple PT Areas that use the same array. Pass zero to return a column number from an array that is only defined for one current PrintTwist area
$2	*ColumnArray*	Pointer	->	Pointer to an array/Field that is a column in a PrintTwist Area

PT_GetColNum is used to find the column number for a ColumnArray. One style of programming that you may use, instead of configuring items in a PrintTwist area by column number, is to configure them by Column Array instead. This way, if you later decide to add or remove columns from your report, you will not have to re-write all the code that is hard coded to column numbers.

Example:

PT_SetFormat(MyArea;***PT_GetColNum***(0;->aSalary);"$###,##0.00";2;0)

PT_GetColumnArray

PT_GetColumnArray(AreaRef;ColumnNumber;BitwiseOptions) -> ColumnArray

#	**Parameter**	**Type**	**In/out**	**Description**
$0	*ColumnArray*	Pointer	<-	A pointer to the array for column ColumnNumber
$1	*ColumnArray*	Pointer	->	Pointer to an array that is a column in a PrintTwist Area
$2	*ColumnNumber*	Longint	->	The column number for which you want to find the array name
$3	*BitwiseOptions*	Longint	->	+1: If this columns is a field, then return a pointer to the hidden array that stores data for this field: else, return a pointer to the field

PT_GetColumnArray is used to return a pointer to the ColumnArray for a specific Column Number.

This function is especially handy when using CallBack methods, which pass an AreaRef and a ColumnNumber to a method that you specify. Using ***PT_GetColumnArray***, you can easily discern the column array from the column number.

Example:

```
`MyBreakFunctionCallbackMethod
c_longint($1;$2;$4;$5)` BreakLevel, ColumnNumber, StartRow, EndRow
c_text($3)
$iStartRow:=$4
$iEndRow:=$5
$Format:=$3
$pColumnArray:= PT_GetColumnArray(vMyPTArea;$2)
if ( $pColumnArray=(->aSalesTerritory))
  $Total:=0
  for($iRowCtr; $iStartRow; $iEndRow)`Start Row to End Row
    $Total:=$Total+aSalesAmount{$iRowCtr}
  end for
  $0:=string($Total/($iEndRow-$iStartRow); $Format)
end if
```

PT_SetFormatPtr

PT_SetFormatPtr
(AreaRef;Format;ColumnJustify;HeaderJustify;UserPictureHeight;->ColumnArray1 {;->ColumnArrayN})

#	Parameter	Type	In/out	Description
$1	AreaRef	Longint	->	PT Area Reference
$2	Format	Text	->	Format for the column(s)
$3	ColumnJustify	Longint	->	Justification for the Column: 0 = Default 1 = Left 2 = Center 3 = Right
$4	HeaderJustify	Longint	->	Justification for the Column Headers: 0 = Default 1 = Left 2 = Center 3 = Right
$5	UsePictureHeight	Longint	->	This parameter is currently ignored by PrintTwist™. See the PrintList™ Pro user's manual for more info.
$6-$N	ColumnArray	Pointer	->	Pointer(s) the column(2) for which you are specifying a format

PT_SetMiscOpts

PT_SetMiscOpts (AreaRef;Opts)

#	Parameter Name	Type	In/out	Description
$1	AreaRef	Longint	->	PT Area Reference
$2	ConstrictRightMostColWidth	Longint	->	If specified, constrict the column width of the right-most column to the specified width. 0 = Behave like standard PLP, and allow the right-most column to extend to the right side of the PLP area. 1 = Constrict the right-most column to it's configured width.

PT_SetMiscOpts is used to set various options and behaviors of PrintTwist™, in ways that differ from PrintList™.

ConstrictRightMostColWidth: With PLP, the right most column ignores the width that the developer specifies, and takes on the remaining available width of the PrintList™ area. While this is often a desirable feature, there are some cases where this is not desired. Right justified columns, especially are often NOT a column that you want to have extended all the way to the right side of an area, especially if this leaves a large gap between columns.

Pass a 1 in ConstrictRightMostColWidth to constrict the right-most column to it's configured width.

PT_GetPageCount

PT_GetPageCount *-> PageCount*

Use ***PT_GetPageCount*** to get a count of the number of pages in the report, **before printing**. This is useful when you want to print a page count on each of the pages of the report.

PT_GetPageCount requires no parameters, and returns a longinteger.

Call ***PT_GetPageCount*** immediately prior to ***PT_Print***. If you use any other PrintTwist command between ***PT_GetPageCount*** and ***PT_Print***, you risk invalidating the page count.

The example database provided with PrintTwist demonstrates how to use PT_GetPageCount.

Example:

```
$iPTArea:=PT_GetAreaRef ("PT_Area1")
PT_SetArraysNam ($iPTArea;1;3;”aInvClientName”;”aInvDate”;”aInvTotal”)
PT_SetHeaders ($iPTArea;1;3;"Client Name";"Invoice Date";"Total")
PT_SetHostForm (->[MyTable];"PT_Main1";50;650;650;700)

VTotalPageCount:=PT_GetPageCount

PT_Print     `Print the report
`End of Method
```

Then, in the form method of you host form, you should have something Like:

```
C_LONGINT(vFromPixel;vToPixel;vCurrentPageNum)
PT_HostFormMethod (->vFromPixel;->vToPixel;->vCurrentPageNum)
vPrintingPage:="Page "+String(vCurrentPageNum)+" of
"+String(vTotalPageCount)
```

PT_Print

PT_Print ({Options})

#	Parameter	Type	In/out	Description
$1	Options	Longint	->	Optional: +1: ClearArraysOption: If you select this option, PT will size all arrays from all reports to zero as it finishes printing each PT Area.

PT_Print is final command in creating a PrintTwist™ report. ***PT_Print*** invokes the PrintTwist™ printing engine and prints the report.

PT_SetWidths2

PT_SetWidths2 (AreaRef;StartingColumn;ScalingFactor;TextWidths{;Width1-N})

#	Parameter	Type	In/out	Description
$1	AreaRef	Longint	->	PT Area Reference
$2	Starting Column	Longint	->	The first column for which you will be defining widths for
$3	Scaling Factor	Real	->	A scaling percentage factor for which to scale all columns in this call of PT_SetWidths2. 100 Is the default. A zero value here will be interpreted as 100.
$4	TextWidths	Text	->	Numeric widths, separated by semicolons
$5-N	Widths	Longint	->	Optional: Column Widths

PT_SetWidths2 is used to specify column widths, very much like ***PT_SetWidths***, just with some increased functionality.

$3: Scaling Factor allows you to scale all columns passed in this call to PT_SetWidths2 larger or smaller. Pass 0 or 100 to keep widths as they are. A value of 50 would make all columns 50% their specified widths, and so on.

You specify the widths EITHER in $4 or in $5-$N. $4 allows you to pass in widths in delimited text.

Examples:
PT_SetWidths2 (vMyArea;1;0;”10;50;50;40”)`Set columns 1 through 4 to widths 10, 50, 50, 40

PT_SetWidths2 (vMyArea;5;75;””;50;50) `Set column 5 & 6 to 50 pixels wide, * 75% (37 pixels wide)

PT_ExecutePLCode

PT_ExecutePLCode (PrintListAreaReference)

#	Parameter	Type	In/out	Description
$1	PrintListArea Ref	Longint	->	PrintList Area Reference

PT_ExecutePLCode is used to apply a PrintTwist configuration that you have programmed to a PrintList™ area, when you are using the when you are using the Delayed Execution wrapper mode of ***PT_GetAreaRef***.

You must call ***PT_ExecutePLCode*** from within the form method of a PrintList™ form. ***PT_ExecutePLCode*** is safe to call from during any form event, but must be called at least once in the ON Printing Detail form event.

PT_SetBrkKeepOnSamePage

PT_SetBrkKeepOnSamePage (PrintTwistAreaRef;BreakLevel;{LastBreakLevel})

#	Parameter	Type	In/out	Description
$1	PrintListAreaRef	Longint	->	PrintList Area Reference
$2	{first}BreakLevel	Longint	->	The break level that you want to configure to keep detail, break header and break footer rows all on the same page.
$3	Last Break Level	Longint	->	If passed, then this is the last break level for which to keep detail, break header and break footer rows all on the same page. PrintTwist will attempt to keep all break levels between $2 and $3 from breaking across pages.

PT_SetBrkKeepOnSamePage is used to tell PrintTwist to attempt to keep break header, detail rows and break footer rows for a specified break level or range of break levels from breaking across pages.

Examples:

PT_ SetBrkKeepOnSamePage (vMyArea;1)`Keep data in the main sort level from breaking across pages

PT_ SetBrkKeepOnSamePage (vMyArea;1;2)`Keep data for break levels 1 and 2 from breaking across pages

Installation

BEFORE YOU INSTALL: IMPORTANT

Before you install PrintTwist into your database, it is important that you first make a backup copy of your database. If anything goes wrong, you want to be able to recover.

The most common problem when installing a 4D component, is that sometimes 4D will corrupt or lose track of certain database objects, such as style sheets or picture library pictures. A good idea is to use 4D insider and get a count of these objects before you install the component, and compare your tally with a count of the same objects after you install. If an objects disappear, then go to a backup of your database and try again.

[PT_Form] table and Linking to a table in your database

The PrintTwist™ component contains one database Table. The sole purpose of this table to contain the forms for PrintTwist™. This table has no fields. It is necessary for any 4D component which has Forms to also contain at least one Table. However, **this table does NOT necessarily need to be installed into your database**. You can decide if you want to install this table, or if you want to “link” PrintTwist™ to an existing table in your database, such as a dialogs table. We recommend that you link PrintTwist™ to a table in your database. This will lead to the least amount of possible issues in the future.

Reinstalling PrintTwist™: IMPORTANT

If you choose to copy the table [PT_Form] into your database the first time you install the component, then, every time that you reinstall the component, you should again choose to copy the table. (Choosing the “Copy” option again doesn't make multiple copies of the table in your database, it just tells insider to use the copy of [PT_Form] that was previously copied with the component.) If you choose to copy [PT_Form] one time, and Link the next time, then 4D Insider will move [PT_Form] out of the component within your database, thus creating a naming conflict for any subsequent reinstallations. If this happens, you MUST rename [PT_Form] to a different name before reinstalling again.

If you choose to install the table [PT_Form], be aware that this table can never be removed from your database. If you uninstall the component, 4D Insider will automatically rename the table to [Deleted table]

If you are reinstalling PrintTwist™, 4D Insider will prompt you for each method that you have modified if you want to update the method with the new method from the component.

Naming Conventions

All objects of the following types are prefixed with "PT_"
Forms, Lists, Named Selections, Picture Library Pictures, Project Methods, Semaphores, Sets, Style Sheets, Tables, Tips

Variables are prefixed with one of two ways:
Arrays are prefixed with "aPT_"
Non-arrays are prefixed with "PT_"

When installing the component into your database, if there are any naming conflicts between PrintTwist™ and your database, 4D Insider will alert you, abort the operation, and log the conflicting items in a log file. In this case, you must rename the items in your database so that they do not conflict with the PrintTwist™ component.

INSTALLATING PrintTwist™ INTO YOUR DATABASE

1. Make a backup of your database structure. (This is recommended before installing any component)

2. Using 4D Insider version 6.8.1 or greater, open your Database. (PrintTwist™ is designed to work in 4D version 6.8.1 or higher databases.)

3. Under the "Component" menu, select "Install/Update"

4. Choose whether you want to
- "Link" PrintTwist™ to a table in your database (This is the recommended option)
- install the [PT_Form] table

5. Close 4D Insider.

6. Install the necessary pluggin into your Mac4DX or Win4DX folder.
   Note: PrintTwist currently requires one of the following pluggins: (either one will do)
   - DC Text2Array (by DataCraft: no-hassle license available)
   - US Count Lines: Part of Foundation Extras. If your application uses Foundation, than you can use this.

   Visit DataCraft on the web at: http://www.DataCraft-Inc.com/

   You must modify the method ***PT_Util_CountLines*** to use your chosen pluggin. To do this, just set the desired branch of the Case statement to True.

Features not yet supported by PrintTwist

The command ***PL_SetSubSelect*** has no supported equivalent in PrintTwist. This command is un-necessary as PrintTwist can print more than 32,000 rows.

Printing of pictures is not yet supported by PrintTwist. We plan to support this in a future version.

Programmatic configuration of the color and thickness of Frame lines and Divider lines is not supported. These ARE customizable by modifying objects on the PrintTwist™ object forms. While the PrintTwist API calls accept the parameters for color and line width of frames and divider lines, the commands do nothing.