# AePlus

# Reporting System API's

SRI Technologies Pty Ltd
WebSite: www.sritech.biz

Email: sritech@sritech.biz

# *Table Of Contents*

# AePlus Reporting System API's

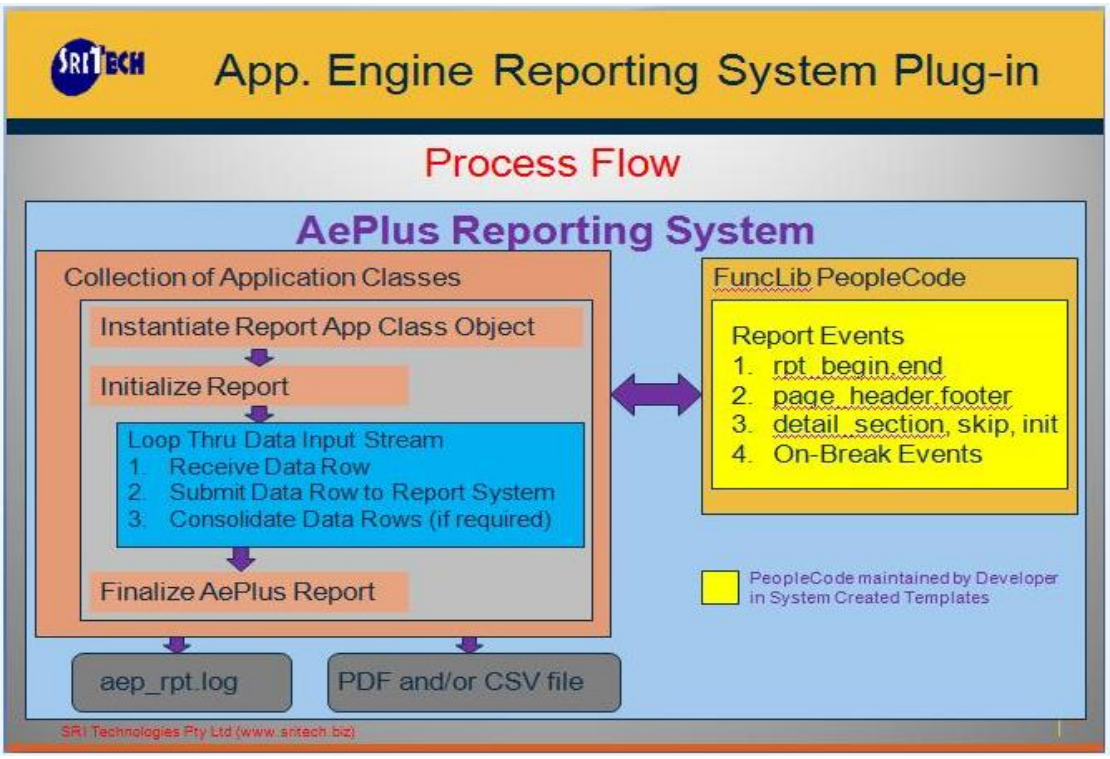
This document describes various AePlus Reporting System API's that developers would call in order to produce PDF reports. Before we start describing these API's, it is important to have a high-level understanding as to how AePlus Reporting System works.

## Reporting System Process Flow:

AePlus reporting system reads a sorted **Input Data Stream** to produce report. It automatically handles various break-points that may occur while processing the rows from Input Data Stream. AePlus Reporting system consists of:

- Delivered Application Classes (AEP_RPT:*).
- An Application Class that is auto generated for given PDF report (e.g. for My First Pdf Report, the application class is AEP_MY_FRST_PDF:*).
- Several FuncLib PeopleCode Functions that are created as template and filled-in/maintained by developer. These PeopleCode contain all Break-Events functions (e.g. PeopleCode functions in AEP_MY_FRST_PDF.AEP_RPT_EVENTS FieldFormula where AEP_MY_FRST_PDF record is referred as ***rpt_aet_rec*** record in AePlus Reporting System Terminology).
- A main PeopleCode body that initiates execution of PDF report. This is generally a PeopleCode step in an Application Engine.
- All above PeopleCode and ***rpt_aet_rec*** record work together to produce professional colourful PDF report.

We will describe various API's in the order developers would use them.



# Primary API's Called By App Engine PeopleCode:

## Report Application Class Instantiation:

Report Application Class Instantiation has the following syntax:

&my_rpt = create AEP_RPT_NAME:AEP_RPT_NAME(&Instance_ArgList);

Where: AEP_RPT_NAME is the report name and &Instance_ArgList contains optional list of arguments/parameters.

e.g. for AEP_MY_FRST_PDF report, PeopleCode will be:

&my_rpt = create AEP_MY_FRST_PDF:AEP_MY_FRST_PDF(&Instance_ArgList);

Above line creates an instance of the report that primarily results in creation of a process log file that will record various messages as the Reporting System progresses. Argument list is mainly used to determine name of the log file and its location. The syntax is:

**abspath**=AbsolutePath,pi=ProcessInstance,appid=AE_APPLID,runctl=RunControl,oprid=OperatorId

Where:

**abspath:** is the absolute folder name where log file and other files will be created by Reporting System. This must end with folder separator character (e.g. '/' for Unix, '\' for Windows). When missing or not terminated by '/' or '\', value defaults to blank and files are created using **%FilePath_Relative** token.

**pi**: is the Process Instance number of the current process. When missing defaults to blank.

**appid**: is the AE_APPLID name of the current process. When missing defaults to blank.

**runctl**: is the Run Control ID for the current process. When missing defaults to blank.

**oprid**: is the Operator ID who is running the current process. When missing defaults to **%OperatorId**.

**All above items are optional but when present the log file name is constructed as:**

{abs/rel path}aep_rpt[_pi][_appid][_runctl]{_oprid}{.log}

It is up to the developer to decide how the log file should be named. Since report can be instantiated thru App Engine PeopleCode or may be thru Push Button PeopleCode, certain items may not be available to specify. The idea is to capture important fields in naming the log file and make it as unique as possible.

If absolute path is missing, reporting system will create all files using **%FilePath_Relative** token and would rely on PeopleTools to make them available in the report repository when viewed thru Process Monitor.

Developer may choose to force the location of log file by using value of PRCSOUTPUTDIR field of PS_CDM_LIST table for given Process Instance. This will also force the location of all other files that Report System produces (like csv, pdf).

## Methods:

There are only 3 primary APIs that developer would use in order to initialise the Reporting System and then Receive and Submit each row of data from **Input Data Stream**.

## Method InitReport()

### InitReport(&ArgListAs string)

This is the first call to Reporting System, where report attributes are defined by passing the list of attributes in &ArgList variable. The list has following format:

**ItemName=***ItemValue*,**ItemName=***ItemValue*,…

List of items are separated by comma and ItemName is case insensitive. Valid items are:

**rpt_aet_rec=**{Record Name That Controls the on-break logic} **- *Mandatory***
**rpt_dat_rec=**{Record Name That holds data from Input Data Stream} **- *Mandatory***
**PageSize=**{Page Size} – Optional Defaulting to A4
**PageOrientation=**{Page Orientation} – Optional Defaulting to P (Portrait)
**FontSize=**{Font Size} – Optional Defaulting to 8
**PageFooterSize=**{Page Footer Size} – Optional Defaulting to 0 (No Footer)
**TopMargin=**{Page Top Margin} – Optional Defaulting to 25 Points
**LeftMargin=**{Page Left Margin} – Optional Defaulting to 25 Points
**RightMargin=**{Page Right Margin} – Optional Defaulting to 25 Points
**BottomMargin=**{Page Bottom Margin} – Optional Defaulting to 25 Points

Above items define two records that report will be based on. In addition, it defines the page attributes of the report.

Example: Following Argument list will set the report page attribute with font size as 10, Page Orientation in Landscape mode with Page footer size as 25.

```
&my_rpt = create AEP_MY_FRST_PDF:AEP_MY_FRST_PDF("");
&ArgList = "rpt_aet_rec= AEP_MY_FRST_PDF";
&ArgList = &ArgList | ",rpt_dat_rec= AEP_MY_FRST_DAT";
&ArgList = &ArgList | ",FontSize=10";
&ArgList = &ArgList | ",PageOrientation=L";
&ArgList = &ArgList | ",PageFooterSize=25";

If &my_rpt.InitReport(&ArgList) = True Then
        …
```

```
        …
End-if;
```

It is important to note that page is always setup with Font Name as Courier. However, Font Size can vary with default being 8. A fixed font is used to setup the page so that Reporting System can work out number of standard lines that a page can accommodate and also number of standard characters that a line can have.

**See Also:**

- my_first_pdf_report_using_peoplecode.pdf tutorial document.
- Method NewPage()

## Method SubmitData()

**SubmitData(&DataRec As Record, &ConsolidateKeyValue As string);**

This is the second call to the Reporting System, where row of data received from **Input Data Stream** is submitted to the reporting system. This call should be within while/for loop that receives one row of data at a time from **Input Data Stream** into &DataRec record variable. Second argument (&ConsolidateKeyValue) is used to indicate if any consolidation is required on input rows. This can be a running number (in string format) If every row received from Input Data Stream should appear on the PDF report (i.e. no consolidation of the rows read from data stream).

Example: Following code shows the data row submitted to the reporting system.

```
If &my_rpt.InitReport(&ArgList) = True Then
        &sel_str = "%SelectAll(:1) a ";
        &SQL = CreateSQL(&sel_str, Record.AEP_MY_FRST_DAT);
        &Rec = CreateRecord(Record.AEP_MY_FRST_DAT);
        &row_count = 0;
        While &SQL.Fetch(&Rec)
                &RowCount = &RowCount + 1;
                &my_rpt.SubmitData(&Rec, string(&RowCount));
        End-While;
        &SQL.Close();
End-if;
```

**See Also:**

- my_first_pdf_report_using_peoplecode.pdf tutorial document.
- Method InitReport()

## Method FinalizeReport()

**FinalizeReport(&Args As string, &DataRec As Record)**

This is the third and last call to Reporting System that initiates generation of ultimate PDF file. &DataRecord has the values that were set in the while/for loop. &Args is the parameter that should be passed and can have following value:

NoDataMsg={Message to be displayed on the report when no data row qualified for reporting}

Example: Following code shows the data row submitted to the reporting system.

```
If &my_rpt.InitReport(&ArgList) = True Then
        &sel_str = "%SelectAll(:1) a ";
        &SQL = CreateSQL(&sel_str, Record.AEP_MY_FRST_DAT);
        &Rec = CreateRecord(Record.AEP_MY_FRST_DAT);
        &row_count = 0;
        While &SQL.Fetch(&Rec)
                &RowCount = &RowCount + 1;
                &my_rpt.SubmitData(&Rec, string(&RowCount));
        End-While;
        &SQL.Close();
        &my_rpt.FinalizeReport("NoDataMsg=*** No Data to Report ***", &Rec);
End-if;
```

**See Also:**

- my_first_pdf_report_using_peoplecode.pdf tutorial document.
- Method SubmitData()

## Other Properties/Methods

In addition to above three API's, there are few supporting Properties and Methods that may be required depending on specific developer's needs.

**property boolean Pdf;**
**property boolean Csv;**

Above two properties dictates to Reporting System if pdf/csv file should be produced. By default system produces PDF file but not CSV file. If you want to produce CSV file, following PeopleCode will be required:

```
If &my_rpt.InitReport(&ArgList) = True Then
        &my_rpt.Csv = True;
        …
        …
End-if;
```

**See Also:**

- my_first_pdf_report_using_peoplecode.pdf tutorial document.
- Method CsvHeader()

**method LoadCustFont(&FontFileNameInp As string) Returns boolean;**

Reporting System supports standard PDF fonts. Developers can use custom TrueType fonts. These fonts should be loaded first before use. Following PeopleCode will be required to use custom font:

```
If &my_rpt.InitReport(&ArgList) = True Then
        &my_rpt.Csv = True;
        &Status = &my_rpt.LoadCustFont("C:\Windows\Fonts\Arial.ttf");
```

```
        ...
        ...
End-if;
```

**property boolean FooterOnLastPage;**

By default, rpt_page_footer() function is not called for the last page of the report. This property can be used to enable it.

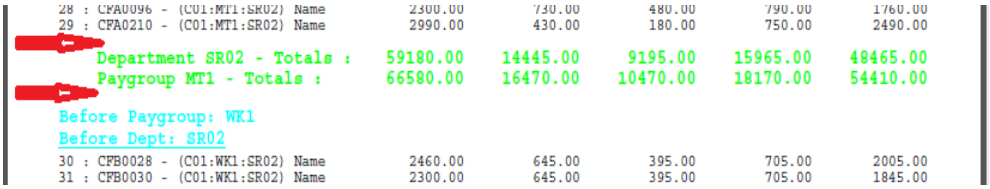
```
If &my_rpt.InitReport(&ArgList) = True Then
        &my_rpt. FooterOnLastPage = True;
        ...
        ...
End-if;
```

**property boolean newline_Before_Beforebreak;**
**property boolean newline_After_Beforebreak;**
**property boolean newline_Before_AfterBreak;**
**property boolean newline_After_AfterBreak;**

Reporting System can print a blank line when processing on-break before/after events. The default values for these properties are:

newline_Before_Beforebreak = False;
newline_After_Beforebreak = False;
newline_Before_AfterBreak = True;
newline_After_AfterBreak = True;

Above values produce output like (blank line shown in Red Arrow):




This default behaviour can be changed if developer's wishes. Following PeoplCode will not produce any blank line while process on-break events.

```
If &my_rpt.InitReport(&ArgList) = True Then
        &my_rpt.newline_Before_AfterBreak = false;
        &my_rpt.newline_After_AfterBreak = false;
        ...
        ...
End-if;
```

**property boolean HeaderOff**
**property boolean FooterOff**

By default Header and Footer event functions will be invoked whenever NewPage event triggers. These functions can be disabled by using these properties.

## API's Called By Report Events (Func Lib) PeopleCode:

This section describes various API's that developers will be using in various Report or On-Break Event functions. These will be essentially related to various Print methods or print positioning on the current page. These functions refer to Print Attributes (e.g. font, font size, colour etc). Printing can be:

**Relative**- MethodNewLine() followed Method PrintStr().

Or

**Absolute** – other Print methods described in this section.

Developer can choose to mix them up. NewLine() method will trigger a New page if current page is full. Once a new page is initiated, any further printing (Relative or Absolute) will happen on the new page and old page becomes inaccessible.

## Print API's Called By Report Events (Func Lib) PeopleCode:

### Method PrintStr();

### Method PrintStrAbs();

**PrintStr(&ColPos As number, &ValueStr As string, &PrintAtrInp As string);**

**PrintStrAbs(&PointX As number, &PointY As number, &ValueStr As string, &PrintAtrInp As string);**

**&ColPos:** Absolute Column Position on the current print line. The value should be in the range of 1..MaxChars allowed in Line.

**&PointX:** Absolute x-Coordinate on the current page. The unit is in Points and valid range is determined by get_PageTopLeft_x()..get_PageBottomRight_x() methods.

**&PointY:** Absolute y-Coordinate on the current page. The unit is in Points and valid range determined by get_PageTopLeft_y() .. get_PageBottomRight_y() methods.

**&ValueStr:** Value to be printed. This is always string data type. If other data types need to be printed (like number, date etc), these should be converted into string using necessary formatting and appropriate PeopleCode Function.

**&PrintAtrInp:** Print Attribute associated with this print string. Following comma separated attributes may be specified.

**FontName={Font Name}**
e.g. Courier, Helvetica, Times-Roman etc. Default is Courier.
**FontSize={Font Size}**
e.g. 10, 12 etc. Default is 8.
**Color={ColorSpec}**
e.g.
magenta, red etc.
Colour number (16711680 for red)
rgb(255,0,0) for red

Following PeopleCode produce identical results:

```
&AEP_RPT.PrintStr(1, "*** End of Report ***", "color=red");
&AEP_RPT.PrintStr(1, "*** End of Report ***", "color=16711680");
&AEP_RPT.PrintStr(1, "*** End of Report ***", "color=rgb(255,0,0)");
```

**centre={Col2_Position}**

Applicable to PrintStr() method only. This attributes prints the string centring between &colPosand Col2_Position. If Col2_position is missing, it defaults to right most column position of the line.

**Right**

Applicable to PrintStr() method only. This attributes prints the string right justifying at &colPos.

**Bold**

Prints string in bold. Applicable to only Standard PDF Fonts.

**Italic**

Prints string in italic. Applicable to only Standard PDF Fonts.

**Underline**

Underlines the printed text.

**csv=n**

Applicable for CSV file. Prints string and also sends it to CSV file at CSV column n.

**csvonly=n**

Applicable for CSV file. Does not Prints string but sends it to CSV file at CSV column n.

**Example:**

```
&AEP_RPT.PrintStr(2, "Contact:", "FontName=Helvetica,FontSize=20,Bold");
```

**See Also:**

my_first_pdf_report_using_peoplecode.pdf tutorial document.

## Method PrintImage();

**PrintImage(&PointX As number, &PointY As number, &Scale As number, &FileName As string);**

**&PointX:** Absolute x-Coordinate on the current page. The unit is in Points and valid range determined by get_PageTopLeft_x()..get_PageBottomRight_x() methods.

**&PointY:** Absolute y-Coordinate on the current page. The unit is in Points and valid range is determined by get_PageTopLeft_y() .. get_PageBottomRight_y() methods.

**&Scale:** Scaling factor applied to image. To reduce the image by 20%, value 0.2 should be passed.

**&FileName:** Full file name specification of the image file. Supported image types are jpg, png and bmp.

**Example:**

```
&AEP_RPT.PrintImage(&xPoint, &yPoint, 0.2, "d:\sqrplus\docs\peoplesoft.jpg");
```

### Method DrawLine();

**DrawLine(&PointX As number, &PointY As number, &PointX2 As number, &PointY2 As number, &PrintAtrInp As string);**

### Method DrawBox();

**DrawBox(&PointX As number, &PointY As number, &PointX2 As number, &PointY2 As number, &PrintAtrInp As string);**

**&PointX:** Absolute x-Coordinate on the current page. The unit is in Points and valid range is determined by get_PageTopLeft_x()..get_PageBottomRight_x() methods.

**&PointY:** Absolute y-Coordinate on the current page. The unit is in Points and valid range is determined by get_PageTopLeft_y() .. get_PageBottomRight_y() methods.

**&PointX2:** Absolute x-Coordinate on the current page. The unit is in Points and valid range is determined by get_PageTopLeft_x()..get_PageBottomRight_x() methods.

**&PointY2:** Absolute y-Coordinate on the current page. The unit is in Points and valid range is determined by get_PageTopLeft_y() .. get_PageBottomRight_y() methods.

**&PrintAtrInp:**

Print attribute for printing the line. Valid values are:

Color={Colour Specification}
LineThickness={Line Thickness} – e.g. , 0,1,2

Print attribute for printing the box. Valid values are:

Color={Colour Specification})
LineThickness={Line Thickness} – e.g. , 0,1,2
fillcolor={Colour Spec}

**Example:**

```
&AEP_RPT.DrawLine(&xPoint, &yPoint, &xPoint2, &yPoint2, "lineThickness=0");
..
..
&AEP_RPT.DrawBox(&xPoint,&yPoint,&xPoint2,&yPoint2, "lineThickness=0,
fillcolor=lightGray");
..
```

### Method PrintTotalPages()

**PrintTotalPages(&PrintAtrInp);**

This method prints total number of pages in the report. Printing is relative to current print position on the page.

**Example:**

```
..
```

```
&AEP_RPT.PrintStr(75,"Page:" | &AEP_RPT.get_PageNum() | " Of ", "Bold");
&AEP_RPT.PrintTotalPages("Bold");
..
```

## Method SetDefaultPrintAttributes();

**SetDefaultPrintAttributes(&PrintAtrInp As string);**

Sets default print attribute. The default print attributes is applicable to only PrintStr(), PrintStrAbs() and PrintTotalPages() methods. The default attributes remains in-tact till method is called again. However, any print attribute specified in Print methods take precedence over default print attribute.

**Example:**

```
..
&AEP_RPT.SetDefaultPrintAttributes("FontName=Helvetica,FontSize=8,color=red");
..
```

## Method PrintStrWrap()

**PrintStrWrap(&bTrial As boolean, &DataRec As Record, &ColPos1 As number, &ColPos2 As number, &ValueStrInp As string, &PrintAtrInp As string, &ContChar As string) Returns number;**

Prints text within specified column positions wrapping the text when necessary. Wrapping splits the text at blank character.

**&bTrial:** Calls the method in trial mode where no printing occurs but number of lines over which the text would be printed. This can be used to test if NewPage() event would trigger as developer would like to handle this situation where text must print together on the same page.

**&DataRec:** Current data Record.

**&ColPos1, ColPos2:** Colum Positions within which the text will be printed.

**&ValueStrInp:** Text string to be printed.

**&PrintAtrInp:** Print Attribute as applicable to PrintStr() method.

**&ContChar:** Continuation character (single char) that will be used when one word of the text is too big to be printed in single line.

**Example:**

```
..
&lines = &AEP RPT.PrintStrWrap( False, &Rec, 2, 35, "We have not received any
purchase order and items have been delivered.", &PrintAtr, "-");
..
```

# Print Position API's Called By Report Events (Func Lib) PeopleCode:

## Method NewLine();

### NewLine(&NeedLinesInp As number, &DataRec As Record);

When printing is in relative mode, this method completes current line and advances to next line so that subsequent printing by PrintStr() method happens on the next line. If there is not enough lines on the page, it triggers NewPage() method by completing the current page. Once a page is complete, subsequent printing will occur only on the next page.

**&NeedLinesInp:** Developer can specify number of spare lines they would like to have before this printing takes place so that whatever block of lines they intend to print stays together on the same page.

**Example:**

```
..
&AEP_RPT.NewLine(1, &Rec);
..
```

```
..
&AEP_RPT.NewLine(5, &Rec);
..
```

If page does not have 5 lines left, a NewPage() is triggered and subsequent printing happens on next page.

## Method NewPage()

### NewPage(&DataRec As Record, &PageAttributes As string);

This method is invoked automatically when printing is in relative mode (i.e. printing is done by using NewLine() and PrintStr() methods only). This should be invoked by developer if printing on the page is in Absolute mode or developers wants to force a new page depending on certain data conditions.

**&PageAttributes:** Page attribute can vary from one page to other. This attributes should be used when current page attributes require a change. Valid values are (but specify only what attribute is changing):

**PageSize**={Page Size}
A0-A6, LETTER
**PageOrientation**={Page Orientation}
P, L
**FontSize**={FontSize}
10, 12 etc
**PageFooterSize**={Footer Size in Points, default 0}
**LeftMargin**={LeftMargin in Points, default 25}
**RightMargin**={Right Marging in Points, default 25}
**TopMargin**={Top Margin in Points, default 25}
**BottomMargin**={Bottom Margin in Points, default 25}
**HeaderOnOff**={Page Header function call, 1 – enable, 0 – Disable}

**FooterOnOff**={Page Footer function call, 1- enable, 0- Disable}
**NewFile**={FileName;resetpagenum}

Start new page with new file name.

FileName (Optional) – use this as new file name. Filename is without path and extension.
resetpagenum (Optional) – reset the page number for new file.

e.g. NewFile=; will use existing filename to derive new file name by appending a number. It will not reset the page number and page numbers will not start from 1 for every new file.

**Example:**

```
..
&file = "company " | &Rec.GetField(Field.AEP_COMPANY).Value;
&AEP_RPT.NewPage(&Rec, "PageSize=A4, FontSize=10, NewFile=" | & file | ";");
..
```

### Method get_CurrPointY();

**get_CurrPointY() Returns number;**

This method returns current Y Position on the current page in Points.

### Method get_CurrLineNumber()

**get_CurrLineNumber() Returns number;**

This method returns current line number on the current page.

### Method get_LinesInPage()

**get_LinesInPage() Returns number;**

This method returns number of standard lines that current page can accommodate.

### Method get_LinesRemainingInPage()

**get_LinesRemainingInPage() Returns number;**

This method returns number of standard lines remaining on current page.

### Method LastReportPage();

**LastReportPage() Returns boolean;**

This method returns true if current page is the last page of the report. Should only be used in Page Footer Event PeopleCode function.

### Method AdjustCurrPointY()

**AdjustCurrPointY(&AdjustPoints As number);**

This method can be used to adjust (change) the current Y position on the page. This may be used when there are minor adjustments required due to varying fonts and font sizes used while printing lines on the report.

### Method AdjustCurrPointY_ByLines()

**AdjustCurrPointY_ByLines(&Lines As number);**

Same as **AdjustCurrPointY** but adjustments are made by passing number of lines instead of number of Points.

### Method SetCurrPointY_AtLine()

**SetCurrPointY_AtLine(&Lines As number);**

This method sets Current Y Point that corresponds to absolute line number on the page.

### Method ColPositionToPoint_X()

**ColPositionToPoint_X(&ColPos As number) Returns integer;**

This method returns Current X Point that corresponds to absolute column number on the page.

## Page Attributes API's Called By Report Events (Func Lib) PeopleCode:

### Method get_CurrPageSize()

**Method get_CurrPageSize() Returns string;**

This method returns current page size (e.g A4, A3 etc).

### Method get_CurrPageOrientation()

**Method get_CurrPageOrientation() Returns string;**

This method returns current page orientation (e.g. P, L).

### Method get_CurrPageWidth()

**Method get_CurrPageWidth() Returns number;**

This method returns current page width in points.

### Method get_CurrPageHeight()

**Method get_CurrPageHeight() Returns number;**

This method returns current page height in points.

### Method get_CurrLeftMargin()

**Method get_CurrLeftMargin() Returns number;**

This method returns current page left margin in points.

### Method get_CurrRightMargin()

**Method get_CurrRightMargin() Returns number;**

This method returns current page right margin in points.

### Method get_CurrTopMargin()

**Method get_CurrTopMargin() Returns number;**

This method returns current page top margin in points.

### Method get_CurrBottomMargin()

**Method get_CurrBottomMargin() Returns number;**

This method returns current page bottom margin in points.

### Method get_FontSize()

**Method get_FontSize() Returns number;**

This method returns standard font size for the current page.

### Method get_CharWidth()

**Method get_CharWidth() Returns float;**

This method returns standard character width for the current page.

### Method get_CharHeight();

**Method get_CharHeight() Returns float;**

This method returns standard character height for the current page.

### Method get_PageTopLeft_x()

**Method get_PageTopLeft_x() Returns number;**

### Method get_PageTopLeft_y()

**Method get_PageTopLeft_y() Returns number;**

### Method get_PageBottomRight_x()

**Method get_PageBottomRight_x() Returns number;**

### Method get_PageBottomRight_y()

**Method get_PageBottomRight_y() Returns number;**

### Method get_PageFooterSize()

**Method get_PageFooterSize() Returns number;**

Above methods return points that define the print area of the current page.

## Other API's:

### Property ReportName

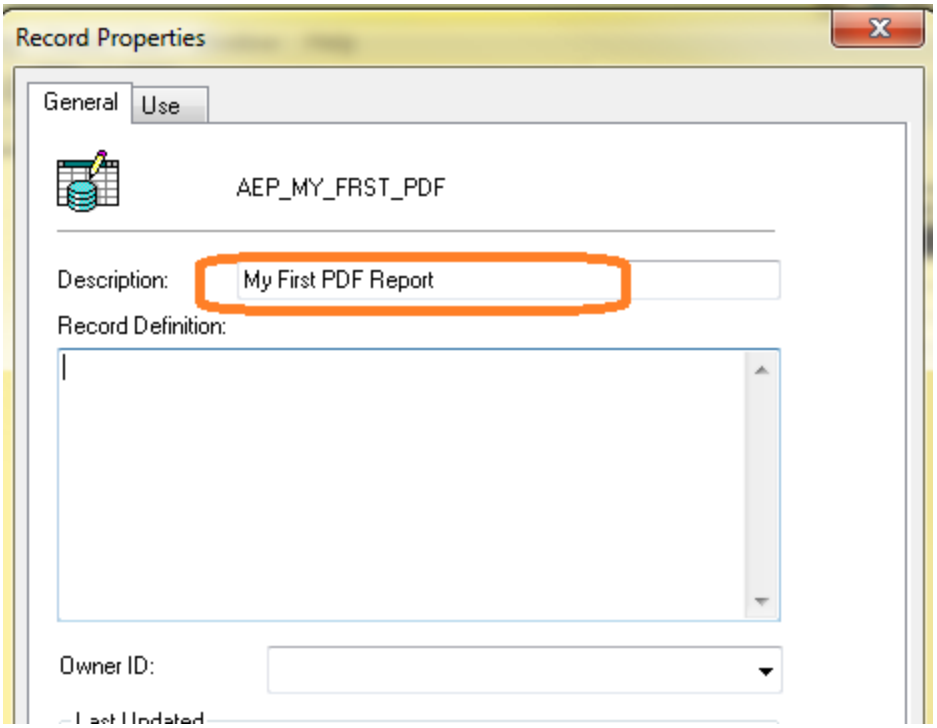
**property string ReportName;**

These property returns report name that is same as ***rpt_aet_rec*** record.

### Property ReportDescr;

**property string ReportDescr;**

These property returns report description that is specified by the developer and can be used while printing the page header.




### Method SetPdfFileName()

**Method SetPdfFileName(&PdfFileNameInp As string);**

Initial PDF output filename is derived from ***rpt_aet_rec***. This initial name can be changed by "NewFile=;" page attribute. SetPdfFileName() method overrides the previously PDF file name. The method can be called any time but before completing the last page of the current PDF file.

For Example, you can use this method when you want to print invoice where each invoice is a PDF file by itself with file name having Customer ID and invoice number.

### Method DataReported()

**Method DataReported() Returns boolean;**

This method returns true if **rpt_detail_section** was called at least once meaning something was printed in the report.

### Method CsvHeader()

**CsvHeader(&CsvDelimInp As string, &ArgsInp As string);**

This method sets the header row of the CSV file.

**Example:**

```
Function rpt_begin(&AEP_RPT As AEP_RPT_DEMO:AEP_RPT_DEMO, &Rec As Record);

   Local string &CsvHeaderStr;
   Local string &Delim = ","; /* comma delimited */
rem Local string &Delim = Char(9); /* tab delimited */

rem setup header live for CSV file;
&CsvHeaderStr = "";
&CsvHeaderStr = &CsvHeaderStr | "SlNo" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Company" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Pay Group" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Department" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "EmplID" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Name" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Basic" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Allowances" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Deductions" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Tax" | &Delim;
&CsvHeaderStr = &CsvHeaderStr | "Net";

&AEP_RPT.CsvHeader(&Delim, &CsvHeaderStr);

End-Function;
```

### Method get_RptRowNum();

**get_RptRowNum() Returns number;**

This method returns row number that is being printed in **rpt_detail_section** function.

### method get_InputRowNum()

**get_InputRowNum() Returns number;**

This method returns row number received from **Input Data Stream**.

### method get_PageNum()

**get_PageNum() Returns number;**

This method returns current page number and can be used while printing page header.

**Example:**

```
..
&AEP_RPT.PrintStr(75,"Page:" | &AEP_RPT.get_PageNum(), "Bold");
..
```

### method ToLog()

**ToLog(&Severity As string, &Message As string);**

This method is a generic logging method that can be used to send a message to aep_rpt.log file that AePlus Reporting System produces.

**&Severity:** Severity of message. Valid Values are:

"D" – Debug Message

"W" – Warning Message

"E" – Error Message

"F" – Fatal Error Message

" " – Informational Message

**&Message:** Text message to log.

**Example:**

```
Function rpt skip(&AEP_RPT As AEP_RPT_DEMO:AEP_RPT_DEMO, &Rec As Record)
Returns boolean;

   Local boolean&RtnSts;

&RtnSts = False;

   If &gbl tax< 700 Or
         &gbl net< 1500 Then
      &RtnSts = True;
   End-If;

   If &RtnSts = True Then
      If &gbl tax>= 700 And
            &gbl net< 1500 Then
        &AEP RPT.ToLog("", "Employee Skipped due to insufficient Net, Emplid: "
| &Rec.GetField(Field.AEP_KEY_DETAIL).Value | ", Tax: " | &gbl_tax | ", Net: "
| &gbl net);
      End-If;
   End-If;

   Return &RtnSts;

End-Function;
```

## Report Events, On-Break Functions:

The Reporting System workbench creates Templates for various FieldFormula functions in ***rpt_aet_rec*** (e.g. AEP_MY_FRST_PDF.AEP_RPT_EVENTS). These functions can be categorised as:

### Report Events:

**rpt_begin(…)**
**rpt_end(…)**
These functions are called once by the AePlus Reporting System. Developer can put necessary PeopleCode against each such Event.

**See Also:**
my_first_pdf_report_using_peoplecode.pdf tutorial document

### Page Events:

**rpt_page_header(…)**
**rpt_page_footer(…)**
These functions are called once by the AePlus Reporting System for each page. Developer can put necessary PeopleCode against each such Event.

**See Also:**
my_first_pdf_report_using_peoplecode.pdf tutorial document

### On-Break Events:

**rpt_before_aep_{*fieldname1*}, rpt_after_aep_{*fieldname1*}**
**rpt_before_aep_{*fieldname2*}, rpt_after_aep_{*fieldname2*},**
**rpt_before_aep_{*fieldname3*}, rpt_after_aep_{*fieldname3*}**
**…**
**…**

These functions are called every time there is a change in ***key*** data value (Fieldname1, Fieldname2, Fieldname3…) when row is received from **Input Data Stream**.

**See Also:**
my_first_pdf_report_using_peoplecode.pdf tutorial document

### Report Data Events:

**rpt_detail_section(…)**
Print row (or consolidated row) received from **Input Data Stream** that should appear in report variables.
**rpt_skip(…)**
Test row (or consolidate row) if it should appear in the Report.
**rpt_init_variables(…)**
Initialize variables that involved in consolidation function.

These functions are called for each row (or consolidated row) received from **Input Data Stream** that should appear on the report.

**See Also:**

These functions are maintained by developer. AePlus Reporting Systems calls them as needed to generate PDF report.

## Page Size and Page Orientations:

AePlus Reporting system supports variety of Page Sizes (A0-A6, LETTER). It supports Portrait and Landscape page orientations.

AePlus Reporting System defaults to following Page Attributes:

| | |
|---|---|
| Page Size: | A4 |
| Page Width: | 595.3 points |
| Page Height: | 841.9 points |
| Page Orientation: | Portrait |
| Font Name: | Courier |
| Font Size: | 8 |
| Left Margin: | 25 points |
| Top Margin: | 25 points |
| Right Margin: | 25 points |
| Bottom Margin: | 25 points |
| Page Footer Size: | 25 Points |

The print area is determined by subtracting margins and footer size from the physical page size. This implies that there is no special allocation for Page Header and can vary depending on how developer uses it. These defaults can be overridden either initially in InitReport() method or later using NewPage() method.

A page in the report can be a mix of any page sizes. For example, you can start the report with page size A4 - Portrait and then change it to A4 - Landscape and then to A3 - Portrait and back to A4 - Portrait.

AePlus reporting system automatically adjusts attributes of the page whenever page is altered so that print contents on the page can be adjusted using current attributes of the page thru PeopleCode.

The default font for the page is always Courier (fixed character size). Based on font size, System calculates number of standard lines that can be printed on the page. It also calculates number of standard characters that a print line can accommodate. Following lines are shown in the process log file (aep_rpt.log) that system generates:

| | |
|---|---|
| Page Font Size: | 8 |
| Char Width: | 5.904 Points |
| Char Height: | 8.440 Points |
| Standard Chars in Line: | 92 |
| Standard Lines in Page: | 86 |
| Page Footer Size: | 25 Points |
| Print Area: | Top Left Corner of page: (25,816) Point, |

Bottom Right Corner of the page: (570,25) Point,
Point to Inch/MM conversion formula: 1 Point = 1/72 inch,
mm = (Point * 25.4) / 72

## Fonts, Sizes and Colours:

AePlus Reporting System supports following standard PDF Fonts.

1. Courier
2. Helvetica
3. Times-Roman
4. Symbol
5. ZapfDingbats

Above fonts along with their variation (Bold and/or Italic) are automatically available for use.

In addition, TrueType font can also be used but these must be loaded first. For example, if you want to use Arial.ttf font, following PeopleCode will be required immediately after InitReport() method. The font is also embedded in the pdf document.

```
If &my_report.InitReport(&InitArgs) = true then
&Status = &my_report.LoadCustFont("C:\Windows\Fonts\Arial.ttf");
    ...
    ...
    While &SQL.Fetch(&Rec)
&my_report.SubmitData(&Rec,...)
        ...
        ...
End-while
&my_report.FinalizeReport("", &Rec,...)
End-if;
```

AePlus Reporting System supports virtually every colour. Following is the list Standard Colours that can be used by name:

1. Black
2. Blue
3. Cyan
4. DarkGrey
5. Grey
6. Green
7. LightGray
8. Magenta
9. Orange
10. Pink
11. Red
12. White
13. Yellow

In addition, colours can be specified by their integer value or by their RGB value. Following three lines of PeopleCode would produce identical results.

```
&AEP_RPT.PrintStr(1, "*** End of Report ***", "color=red");
&AEP_RPT.PrintStr(1, "*** End of Report ***", "color=16711680");
&AEP_RPT.PrintStr(1, "*** End of Report ***", "color=rgb(255,0,0)");
```