```
The following lines show how to embed email commands. Users may need to
set proper values of EmailSMTPServer, EmailSMTPUserName,
EmailSMTPPassword in the printer setting or by embedded commands if
using SMTP mode.

%%EmailEnable: True%%   // True to enable Emailing
%%EmailSendMethod: 0%%  // 0,1: Email client, 2: SMTP
%%EmailTo: customer1@yourcom.com;customer2@yourcom.com%%
%%EmailTo: :;append@appendaddr.com%%
%%EmailCc: antoherone@wherecc.com%%
%%EmailBCc: bcc@wherebcc.com%%
%%EmailFrom: mysales@iteksoft.com%%
%%EmailSubject: PLACE SUBJECT LINE HERE DIRECTLY%%
%%EmailSubject: : concatenate another line%%

%%EmailContent: The first line of content field%%
%%EmailContent: &The second line will be appended with a new line %%
%%EmailContent: &This is achieved by '&' special character. %%
```

Usage Guide for Developers to Control eDocPrinter PDF Pro by Embedded Commands, Registry Settings, and Escape Codes

Last update: 24 Nov 2007

Changes Note:

Ver 6.38 Build 5168 (Nov 24, 2007):

(1) [Doc Info]: Add option Display Doc Info Title for making reader show Title on Window title bar. (DocViewDisplayDocTitle)

(2) [Destination- PostAction]: Add "Delete the destination PDF" in for removing the final PDF without launching the PDF reader. DestActionMode=4

(3) [Redirect2Print]: Add option "Use Reader to Printer" (RedirectUseReader) for calling reader to print without the need of pdj.

(4) [Command2Action- acnt]: New action command acnt is available for adding notes by commands.

(5) [Command2Action- acpf]: The action command acpf is added for allowing processing existing PDF instead of the current PDF. The current job file can be used as pure embedded commands.

(6) [Command2Action- Variables]: Support resolving environment variables and eDocPrinter defined date and time variables (%#DATE%, %#TIME%, %#DATEX%, %#TIMEX%%, %#YYYY%, %#MM%, %#DD%, %#HOUR%, %#MIN%, %#SEC%) in embedded commands DestDir,

DestFile, Filepath, EmailSubject, and EmailContent command. %#FNAME%, %#FTPServer% and %#FTPServerDir% are only available in EmailSubject and EmailContent command.

(7) [Command2Action- Variables in watermark]: Support resolving command2action variables (environment variables and eDocPrinter defined variables) in text watermarks.

(8) [FTP- Ent. pack]: Support FTP for uploading the result PDF. Available commands and registry settings include FTPEnable, FTPServer, FTPUserName, FTPPassword, FTPPortNum, and FTPServerDir.

Ver 6.34 Build 5032 (Jun 9, 2007):

(1) [Destination- Overlay with]: Support using images *.jpg, *.bmp, or *.tif as templates directly in DestOverlayPDFTemplatePath.

(2) [Destination- Overlay with]: Support resolving environment variables in DestOverlayPDFTemplatePath.

(3) [Destination- Filepath]: Support resolving environment variables in Filepath.

(4) [Redirect2Print]: New special printer name "->DEF", "->TIF", and "->TIFG3" are added. "->DEF" is used for redirect to print to the default printer.

(5) [Email]: Add EmailAttachExt, which is available in registry settings and commands. It is used for attaching the same path as destination PDF but with different file extension. For example, it can be used to attach the TIF created by redirect2print, %%EmailAttachExt: .tif%%.

(6) [Embedded Commands]: Add new commands %%ocbdc %%ocemc for adding layers by commands.

Ver 6.30 Build 4782:

(1) [Destination- Overlay]: Add new options for supporting overlay and then append using 2-pages template. These new values 4,5,6,7 in command DestOverlayWithType represent [As Foreground and Appended] [Inserted and As Foreground] [As Background and Appended] or [Inserted and As Background].

(2) [Redirect2Print]: Support the new modes 4,5,6,7 of DestOverlayWithType and DestOverlayType in Redirect printing.

(3) [Shell Extension]: Add support of customization of Windows Explorer Shell Extension right-click shortcuts.

(4) [PDF/A-1b]: New registry settings for setting the printer profile for PDF/A-1b compliant PDF creation.

(5) [Printer Server]: New registry settings for supporting the printer server mode. (It requires per-server Intranet Server License)

Ver 6.24 Build 4398:

(1) [Watermark]: WatermarkEnable and WatermarkSelected are available as embedded commands to enable stamping watermark and selecting watermark

to stamp. Administrators can define global watermarks and stamp mandatory watermark by watermarks in HKLM.

(2) [Command2Action- Watermark]: New advanced commands %%acim:, %%acit, and %%aciw for dynamically defining image, text watermarks, or selecting and stamping multiple watermarks with different attributes or page ranges.

(3) [Command2Action- Form Fields]: New advanced command set %%acft, %%acfb, %%acfk, %%acfr, %%acfl, %%acfs for adding text fields, button, checkbox, radio group, list box form fields by embedded commands directly. Refer developer's commands usage guide for details.

(4) [Command2Action- Variables]: Support adding variables in text watermarks. Two variables #p1and #p0 are supported for representing current page number and total pages.

(5) [Email]: [EmailAttachSelf], default value True, is available for allowing attaching the current PDF when emailing.

(6) [Special Registry- RemoteRegServer]: Support assigning RemoteRegSever setting, e.g (\\server1), for reading global default and MUST settings from remote server HKLM instead of local machine HKLM.

(7) [Com2Control Addin]: A COM Object exposed and registered as "eDocPDFAddin.Ctrl" for developers to control eDocPrinter registry settings easily.

Ver 6.20 Build 4268:

(1) [Destination]: Add new command and registry settings [DestSplitJob] for simplifying [JobSplitPDF] usage. Users can assign as number of pages for splitting on every fixed number of pages.

(2) [Destination- Synchronization]: Add command and registry setting [DestWaitPDFReadyMsg]. When the PDF creation is fully finished, it will broadcast this message name registered. It provides an alternative solution to event based [JobPDFReadyEvent].

(3) [Destination- Overlay]: Add support for [DestOverlayShowWhenDisplay] and [DestOverlayShowWhenPrint] for showing the layer. These are available when creating layer PDF is enabled.

(4) [Destination- Overlay]: Add support “Appended” and “Insert Before” to each page when processing template pages in addition to overlaying as background or foreground. The corresponding commands are [DestOverlayWithType] and [DestOverlayType] with values 0/1/2/3.

(5) [Destination- Overlay]: Add support for [DestOverlayRepeatInterval] and [DestOverlayBackCoverNum] for controlling Even/Odd or 1-2-3 overlaying, and allowing the back-cover containing multiple pages.

Ver 6.18 Build 4060:

(1) [Command2Action Addin]: New advanced embedded command [acef] for embedding files into PDF directly. For example,
%%acef: filepath[,iconstyle,[R,G,B]]%%

(2) [Email2Notes]: Supporting sending email by Lotus Notes when [EmailSendMethod] is 3. In this mode, it also supports sending as HTML, asking for return receipt, and importance level.
(3) [Email]: Add [EmailReturnReceipt] and [EmailImportance] for supporting asking return receipt and setting importance level. Add [EmailSendNow] for controlling trigger Outlook to send email immediately. EmailSendNow is ONLY effective when EmailUseEMAPI and EmailUseOutllok are both True.
(4) [Redirect2Print: Redirect2JPG and Redirect2BMP]: Add two special printer name as "->JPG" and "->BMP" for Redirect2Print addin. When using this special printer name in Redirect2Print , print jobs will be saved as JPG or BMP in the same filename as PDF with auto-numbered and file extension in JPG or BMP.

Ver 6.16 Build 3902:

(1) [Command2Action Addin]: Advanced embedded commands set for supporting adding images, link action, bookmarks, named destinations, and etc. It requires additional Adv. Toolbox license or Adv. Pack License. %%acbk, %%acln, %%acim are available now for adding bookmarks, links, and images.
(2) [EmailAttach]: Support wild card format in EmailAttach command. Users can now assign C:*.pdf or similar pattern to attach multiple files.
(3) [Filepath]: It now will create the directory recursively if the path is not existing.
(4) "True" or "true" will both recognized as "True" for Boolean command values.
(5) [Redirect2Print Addin]: Support redirecting current print job to other printers simultaneously.

Ver 6.14 Build 3666:

(1) UI option [Wait Synchronization of PDF ready event] is controlled by command DestWaitPDFReadyEvent, which is used for synchronizing batch appending into the same destination to avoid race condition.
(2) When UI option [Display SaveAs dialog after print job spooled] is ON (registry setting [DestPostSaveDlg] is True), users can control the DestSaveMode on the fly by embedded commands. Registry setting [DestPostSaveDlg] cannot be controlled by commands on the fly. Just as enabling detecting commands, users have to enable this option in the profile setting (or 6.14 installer may install extra ERP2PDF profile which enables embedded commands and [DestPostSaveDlg] by default.)
(3) [DocViewPrintScaling], [DocViewHideMenubar], [DocViewHideToolbar], and [DocViewHideWindowUI] are new commands and registry settings for controlling reader preferences.

Ver 6.12 Build 3560:

(1) Support sending content in HTML with EMAPI and Outlook. EmailUseEMAPI and EmailUseOutlook are the corresponding commands and registry settings.

(2) EmailFrom is effective in EMAPI mode

Ver 6.10 Build 3390:

(1) Add JobSplitPDF for supporting splitting PDF in the same print job
(2) Add EmailAttach for adding extra multiple documents when emailing
(3) Add FilepathUseDir for supporting assigning a Filepath a pure filename, which will finally be prefixed with the DestDir assigned in Destination property page.
(4) Add EmailSendAsHTML and EmailBodyEncoding for SMTP mode, which enabling send email HTML format. Currently, it is only available in EmailSendMethod: 2 (SMTP mode)
(5) Fix EmailContentFile ill-function in SMTP mode in 6.09

Ver 6.09 Build 3298:

(1) Add a command EmailContentFile for loading email content from a text file.
(2) Add a command DestSignRotateAP for assigning rotate mode when applying signature appearance.
(3) Support 2 new UI options for controlling hiding embedded commands, [Remove text line starting with tag] and [Remove the page with commands only]. It's suitable for using to remove extra pages full of commands.

Since Ver 6.07 Build 3168, eDocPrinter PDF Pro supports dynamically changing security settings by embedded commands on current printing job without updating profiles.

Since Ver 6.07 Build 3168, eDocPrinter PDF Pro supports applying digital signature silently with PDFSealer by embedded commands. (Need PDFSealer 6.02 build 1208 or later)

Since Ver 6.03 Build 2806, eDocPrinter PDF Pro supports embedded commands in normal text strings. This provides a convenient way for automation of controlling eDocPrinter PDF Pro without any special API or registry read/write operations.

The settings for "Embedded Commands" are in the same property page of "Links". Users have to check ON the option "Enable detecting embedded commands" to process commands.

Because the embedded commands are processed after print jobs starting, settings of Email, DocInfo, and Destination (except for those related to prompting SaveAs dialog) can be overridden in the current print job.

In addition to overriding the settings current print job, users can also send special commands to enter "Profile Update" mode. In this mode, the settings will be flushed into registry. And these changes will not be effective until next printing job.

To test embedded commands, please first check ON the option "Enable detecting embedded commands" in the "Link" property page. And print this file into eDocPrinter

PDF Pro. Users may need to set to silent printing mode without prompting SaveAs dialog by the options in Destination property page. Users may take advantage of multiple profiles to set up some specific printer instance dedicated for processing embedded commands.

Embedded Commands

Syntax:

The embedded commands share the similar key-value pair style as registry settings. Users have to specify the key-value pair in %%[alnum]+: [:&][anychar]*%%, where %% is the optional tag for detecting start and end of commands. The tag character can be defined as other character in the property page. The default one is '%'. Repetition of the first character of tag string specified by the user composes the start-tag and end-tag. The commands are case sensitive. Be careful to set commands as the same of its corresponding registry key name.

There are 2 special characters for embedded commands in addition to "start tag" and "end tag". These are ':' and '&'.

':' is used to represent the end of the key, e.g., EmailSubject:
':' is used to represent that the value shall be appended to existing one rather than overwriting the existing values.

Options:

In addition to tag character, there are still other options to control processing embedded commands.

[Delete first page after processing]:
When this option is ON when enabling detecting embedded commands, the final PDF file will not contain the first page of original content. Users then can place all commands in the first page. These commands will not be shown on the final PDF.

[Font Attributes]
The options provide font attribute matching criterion to help users restrict matching conditions. Users can select and enable matching font family, style, size, or color for detecting commands. The matching is automatically disabled when there is no any matching criterion enabled.

When font attribute matching is enabled, users can select the option [Do not show text matching the font attributes] to hide all those text commands.

Since ver 6.10, users can use embedded command [JobCommandFile] for reading commands from a text file instead of embedding directly. However, commands in the text file must be explicitly stated. Unlike used in database report or similar applications, users can use application dependent field names or variables within the command, which will be resolved (or merged, flattened) when report designer print it. For commands in text

file, these commands cannot contain variables or fields of applications. Users have to make sure all commands are resolved (or merged) correctly if the command text file is dynamically generated. [JobCommandFile] is processed on every end of page. Hence it can be used with JobSplitPDF. Users may assign every new command file for every newly split PDF. Suggest users may place fixed commands or default values within this command file without duplicating them every time.

Since Ver 6.16, [Command2Action Addin] - Advanced embedded commands set is added for supporting adding images, link action, bookmarks, named destinations, and etc. It requires additional Adv. Toolbox license or Adv. Pack License.

Email Sample:

For example,
EmailTo: aa1@test1.com;
EmailTo: aa2@test2.com

The second EmailTo: will override the settings of the first one. On the other hand, if users use ':', it will be appended to the existing value. For example:

EmailTo: aa1@test1.com;
EmailTo: :aa2@test2.com

By the prefix ':', the second EmailTo: will be appended. The final result shall look like aa1@test1.com; aa2@test.com

'&' has similar effects, but a new line character '\n' will be added to achieve representing field values with multiple lines. For example, EmailContent: has the need for multi-lines input.

Please refer to the header in the first page for a demonstration on automating email with proper settings by embedded text commands.

Since Ver 6.09 Build 3290, users can use EmailContentFile for loading a text file as email content directly without need to use EmailContent line by line for the whole content.

Since ver 6.10 Build 3390, users use EmailAttach for attaching extra documents in addition to current PDF generated. For example,

```
%%EmailAttach: c:\demo1.gif; c:\logo.gif%%
```

where semicolon is used for separating multiple files.

Since 6.16, users can assign wild card format, e.g., C:*.jpg, for EmailAttach command. Since 6.20, users can use double quote to wrap the filenames containing comma or semicolon to avoid incorrect parsing. For example,

```
%%EmailAttach: "c:\demo,1-1;3.gif"; c:\logo.gif%%
```

Since 6.24, users can set EmailAttachSelf command or registry setting to disable attaching the current PDF created when emailing. This is useful when users want to automate some control flow by printing a document with commands only. The default value is "True" for automatically attaching the current PDF when emailing as earlier versions.

When sending Email in SMTP mode, i.e. [EmailSendMethod: 2], users can enable sending email content in format HTML by adding command [EmailSendAsHTML:

True]. [EmailBodyEncoding] is used for set the HTML encoding, which, by default, is [utf8].

Use Outlook:

Users can enable “Use EMAPI” to support sending by Outlook with support of HTML content, Outlook signature, and silently sending without security warning box with Outlook.

```
%%EmailEnable: True%%                  // Enable emailing
%%EmailSendMethod: 0%%                 // Select sending method
%%EmailUseEMAPI: True%%                // Use E-MAPI (for Outlook)
```

Use Outlook with HTML:

Users have to add the following line for sending emails by Outlook in HTML

```
%%EmailSendAsHTML: True%%                    // Send Email content in HTML
```

Use Outlook with HTML containing images:

Since Ver 6.20,

```
%%EmailUseEMAPI: True%%                      // Use E-MAPI (for Outlook)
%%EmailSendAsHTML: True%%                    // Send Email content in HTML
%%EmailContent: <br><img src="cid:logo.jpg">%%
%%EmailAttach: C:\logo.jpg%%
```

Use Outlook with Signatures or Stationary:

Users have to enable the EmailUseOutlook for automatically appending the Outlook default signature in the email content.

```
%%EmailUseOutlook: True%%              // Read Outlook default signature
```

Since Ver 6.20, when in HTML mode, it will automatically read and embed the images used in the Outlook signature or stationary.

Use Outlook with x64 platform:

In x64 platforms, eDocPrinter PDF Pro is a native 64-bit printer driver. However, Outlook may still be a 32bit application. For calling 32bit EMAPI, users have to enable to call 32-bit client from 64bit drivers. [EmailMAPIThunk32] is the corresponding registry setting of this. Enabling this, it will launch a 32-bit process for calling EMAPI in proper 32-bit context.

On 32-bit Windows, it is not necessary to enable this. However, if users run in batch synchronous mode (DestWaitPDFMode = “1” or “2”) with emailing by Outlook for a lot of jobs, EMAPI may induce and accumulate internal handle leakage depending on various Outlook versions. (Such kind of leakage is caused by EMAPI itself, not from the caller.) Users may enable this to avoid such kind of leakages from EMAPI itself as emailing is done in a separate process for every job.

Use SMTP:

When EmailSendMethod is 2, it will be in SMTP mode. In this mode, it will connect to the SMTP server assigned to send the email. “EmailFrom” field is necessary in SMTP mode. If the SMTP server requires authentication, users have to assign the authentication mode.

```
%%EmailEnable: True%%                // Enable emailing

%%EmailFrom: from@yourcom.com%%
... ...

%%EmailSendMethod: 2%%               // Select sending method SMTP
%%EmailSMTPServer: yourcom.com%%     // SMTP server in IP or name
%%EmailSMTPAuthMode: 1%%             // 0, 1, 2 for different AUTH
%%EmailSMTPUserName: username%%
%%EmailSMTPPassword: password%%      // user name and password for
logging into SMTP account

%%EmailUseHTML: True%%               // Send Email content in HTML
... ...
```

If the SMTP server requires a special port to connect, users can change it by commands:

```
%%EmailSMTPUsingPort: True%%
%%EmailSMTPPortNum: 25%%      // special port number if necessary
```

Use Lotus Notes:

When EmailSendMethod is 3, it will be in Email2Notes mode. It is available since Ver. 6.18. It requires Adv. Pack. (Email2Notes addin will install an extra “epdfnote.dll” in the system) This method is for sending email through LotusNotes silently. Sending email in HTML, asking for return receipt, and set email importance are also available in this mode. This method assumes Notes clients has already installed successfully in the system.

Doc Info Sample:

```
Here shows how to update doc info by embedded commands
%%Title: This is the title of the PDF Document%%
%%Subject: Document Subject in Document Information%%
%%Keywords: eDocPrinter PDF Pro%%
%%Author: ITEKSOFT for testing embedded commands%%
%%DocInfoEnableCustom: True%%       // True to enable custom field

%%CustomDocInfoMyFields1: Customized document field placed here%%
// it means adding a field name “MyField1” with value specified into
// Doc Info of PDF
%%CustomDocInfoMyFields2: extra doc info field%


%%DocOpenActionEnable: True%%       // enable Open Action
%%DocOpenPageNum: 1%%               // jump to specific page number
%%DocOpenPageZoom: 8%%
// zoom to mode 0, 1, 2,…; mode 8 means 100%

%%DocOpenLayoutMode: 0%%            // mode 0, 1, 2, 3
%%DocOpenPageMode: 0%%              // mode 0, 1, 2, 3


%%DocViewPrintScaling: 0%%

%%DocViewHideMenubar: False%%
%%DocViewHideToolbar: False%%
%%DocViewHideWindowUI:False%%


%%DocInfoFilenameAsTitle: True%%
// Since ver 6.16, it will automatically generate title by filename
```

Please refer to the above redline section for demonstration of Document information update.

Destination Sample:

Because embedded commands are processed after printing jobs starting, those options for suppressing the SaveAs dialog must be set in the profile settings initially. We suggest users utilizing multiple-profile feature to set up a printer setting for silently printing. Suggest users to set the SaveAs mode to [Save as using the Name auto-generated with default directory] or [Save as using the Name specified with default directory]. And [When File exists] checking mode should also be set to non-prompting, e.g., [Replace directly]

Though controlling prompting SaveAs dialog must be set initially in the profile settings, most other options of Destination can be overridden by directly embedded commands, e.g., Merge mode, Overlay options, and so on.

```
These options show how to override Destination
Change @@ to %% to make these commands effective.
@@Filepath: C:\test1.pdf@@          //Destination path of the PDF file
@@DestExistMode: 3@@                //3 is [Merge by Append to]
@@DestEnablePostAction: False@@     //Disable launching PDF reader
@@DestRemoveFinalPDF: True@@        //Delete the PDF After Email
```

For some cases, users may want to assign Filepath to a pure filename only, which will then be saved into the directory depending on printer profile setting on different users. Since Ver 6.10 build 3390, users can use, for example,

```
@@Filepath: test1.pdf@@       //relative path of the PDF file
@@FilepathUseDir: True@@      //Use DestDir set in Destination page
```

Of couse, users can also set [DestDir] in commands too, which will override the settings in property page. If [DestiDir] is empty, the PDF file created will be saved to the current directory of the current process.
(Since 6.12, the driver layer will auto-detect filepath is containing the directory or not, if not, it will use DestDir automatically, FilepathUseDir is no use now.

Overlay with:

The following example shows how to enable to overlay the destination PDF with a stationary PDF template. This option is equivalent to the [Overlay with] option shown on the SaveAs dialog and Destination property page.

```
Change @@ to %% to make these commands effective.
@@DestOverlayPDFTemplateEnable: True@@          //Enable [Overlay with]
@@DestOverlayPDFTemplatePath: C:\temp1.pdf@@    //template PDF path
```

Since Ver 6.34, users can assign .jpg, *.bmp, or *.tif as templates directly in command DestOverlayPDFTemplatePath. It also supports resolving environment variables in the path.

Options for [Overlay with] and [Merge by Overlay]:

There is an UI dialog for setting various options for doing “Overlay”. These settings are fully available in registry settings and commands. Users can refer to Appendixes for details of the registry settings.

Overlay method: in PDF

[Overlay with] provides a way for overlaying the PDF created with the pre-generated template PDF. [Merge by Overlay] is similar, but it will use the newly created PDF as the template to overlay its content on the existing PDF.

In Ver 6.18 and earlier versions, users can choose to overlay as background or foreground. This is controlled by [DestOverlayType] with value “0” or “1”. Value “0” will overlay as foreground and value “1” will overlay as background when [Merge by Overlay]. When doing [Overlay with], the effect is reversed. Value “0” will overlay template as background.

Since Ver 6.20 or later, a separate [DestOverlayWithType] is available for controlling “Overlay with” directly. [DestOverlayType] and [DestOverlayWithType] can have values “0/1/2/3”. As the UI selection order, mode “2” and “3” are for putting the template page “Appended” or “Insert Before” for every [DestOverlayRepeatInterval] pages.

Since Ver. 6.30, new values 4,5,6,7, which represent [As Foreground and Appended] [Inserted and As Foreground] [As Background and Appended] or [Inserted and As Background], are added for [DestOverlayWithType] and [DestOverlayType]. These new options support [Overlay and then Append] or similar operations with using a 2-pages template PDF. The first page will be overlaid and the second page will be appended after or inserted before this page.

Template pages structure:

The simplest case is there is one page in the template PDF only. Users can use it as the background stationary for overlaying on each page PDF created.

eDocPrinter PDF Pro provides more flexible ways for doing overlaying. Users can assign to insert header pages and append trailing pages by [DestOverlayPrependNum] and [DestOverlayAppendNum].

If there is more than one page in the template PDF, it will overlay one by one mapping for the template page and destination page like traditional overprinting on used papers. By default, it will repeat the last page of the template page for overlaying. Since Ver

6.20, users can assign [DestOverlayRepeatInterval] for repeating last 2 pages or 3 pages or so for doing even/odd overlaying with different background or similar actions.

Users can combine these options to allow a very flexible template overlaying. For example, overlay first page with cover template, and repeat overlaying even/odd templates, and append 2 trailing pages. In this case, users should prepare a 5 pages template PDF. The first page is the special cover. The following 2 pages are for even/odd stationary, and the last 2 pages are the trailing pages. And set the commands as:

```
@@DestOverlayPDFTemplateEnable: True@@              //Enable [Overlay with]
@@DestOverlayPDFTemplatePath: C:\temp1.pdf@@ //template PDF path

@@DestOverlayWithType: 1@@    // as background

@@DestOverlayPrependNum: 0@@  //
@@DestOverlayAppendNum: 2@@   //append the last 2 pages from the template

@@DestOverlayLastPageIsCover: False@@        // no special back-cover
@@DestOverlayBackCoverNum: 1@@               // if back-cover enabled,

@@DestOverlayRepeatLast: True@@
@@DestOverlayRepeatInterval: 2@@     // repeat last 2 remaining pages

// 1 page left, it will be used as overlaying the first page
```

Layers in PDF:

When overlaying, users can choose to create a new layer or not. This feature requires Reader 6.0 or later. The corresponding registry settings are [DestOverlayNewLayerMergeEnable] and [DestOverlayNewLayerTemplateEnable].

Since Ver 6.20, users can select to allow showing this layer when display or print by the settings of [DestOverlayShowWhenDisplay] and [DestOverlayShowWhenPrint].

Since Ver 6.34, two commands are added for supporting creating PDF layers directly by embedded commands. The printing content wrapped by these two commands will be treated as a PDF layer.

Syntax:
%%ocbdc: layername[,bShowWhenPrint,bShowWhenView]%%
%%ocemc: %%

The first command is used to start a layer. The second is used to close a layer. These commands have to be paired within the same page. The printing content between the two commands will be treated as a layer with layer name and attributes assigned.

These commands assume printing data are sequentially sent to the driver by the application as its visual page layout.

Pseudo code sample:

```
...
        //Draw layer 1
        TextOut(0,0, "%%ocbdc: Layer1%%");
        // draw other shapes in this layer
        TextOut(0,1000, "%%ocemc: %%");


        //Draw layer 2
        pDC->TextOut(0,0, "%%ocbdc: Layer2%%");
        // draw other shapes in this layer
        TextOut(0,1000, "%%ocemc: %%");

        //Draw layer 3
        pDC->TextOut(0,0, "%%ocbdc: Layer3%%");
        // draw other shapes in this layer
        TextOut(0,1000, "%%ocemc: %%");
...
```

Notes:
The layer commands are mainly for developers who write the printing code themselves. For most text based printing applications, the commands may work well as expected when inserted directly with text content. For printing content with graphics/text mixed, if the application does not guarantee the printing order, these commands may not be suitable. For example, some applications may print graphical shapes after all text printed.

For details of available key-value pairs when using embedded commands, please refer to the appendix for registry settings of eDocPrinter PDF Pro. Currently, major key-value pairs in category “Email”, “Doc Info”, “Security”, and “Destination” can be overridden by embedded commands.

Change SaveMode On the fly:

Since 6.14, when UI option [Display SaveAs dialog after print job spooled] is ON (registry setting [DestPostSaveDlg] is True), users can control the DestSaveMode on the fly by embedded commands. The option [DestPostSaveDlg] cannot be controlled by commands on the fly. Just as enabling detecting commands, users have to enable this option in the profile setting directly. Hence administrators can easily disable these features by the profile settings. (In Ver 6.14, the installer may install extra ERP2PDF profile which enables embedded commands and [DestPostSaveDlg] by default.)

When [Display SaveAs dialog after print job spooled] is True,
please refer the following explanation for details of the relationship among Filpath, DestDir, and DestFile. (Users may need to change @ to % in the following examples.)

Case 0:

```
@@DestSaveMode: 0@@
@@Filepath: C:\demodir\demodest.pdf@@
```

In this mode, it will prompt the SaveAs dialog using the value of [Filepath] assigned. If [Filepath] is only a filename only without directory information, it will then use the value of [DestDir] to set the default directory. If [DestDir] is empty

Case 1:

This mode is called: Using the name specified with the default directory.
Hence it will use [DestFile] if it is not empty.

```
@@DestSaveMode: 1@@
@@Filepath: demodest.pdf@@
@@DestFile: @@
```

Basically, it will use [DestDir] and [DestFile] to generate the final destination PDF filename and path. If [DestFile] is empty, the filename in [Filepath] will be used. Users can set [DestFile] and [DestDir] by commands on the fly as well as [DestSaveMode] if [Display SaveAs dialog after print job spooled] is True.

Case 2:

This mode is called: Using the name automatically generated with the default directory.
In this case, users can assign [Filepath] directly, which will override the auto-generated path. If [Filepath] contains only filename without directory information, [DestDir] will be used. If [DestDir] is empty, the current working directory of the application process will be used.

```
@@DestSaveMode: 2@@
@@Filepath: c:\demodir\demodest.pdf@@
```

Security Sample:

Since Ver 6.07 Build 3168, users can directly use embedded commands to dynamically change security settings on current printing job without updating profiles.

```
@@EnableOwnerPasswd: True@@         // Enable owner password
@@EnableUserPasswd: True@@          // Enable user password
// True or False; enable either one will activation encryption


@@EncryptLevel: 40@@          // 40 or 128; encryption bit mode


@@OwnerPasswd: ABCDEF@@       // Owner password
@@UserPasswd: abcdef@@        // User password


// 40 bit mode permission settings
@@E40_NoComment: False@@      // True or False
@@E40_NoCopy: False @@        // True or False
@@E40_NoEdit: False @@        // True or False
@@E40_NoPrint: False @@       // True or False


// 128 bit mode permission settings
@@E128_Access: True@@   // True or False; Enable accessibility support
@@E128_Copy: True@@     // True or False; Enable copy and extraction
@@E128_Edit: 4@@        // 0,1,2,3,4 as the UI, 4 is fully editable
@@E128_Print: 2@@       // 0,1,2 as the UI, 2 is high fidelity printing
```

Digitally Signing Sample:

There are now 6 commands (registry settings) for controlling calling PDFSealer silently to sign PDF created from eDocPrinter PDF Pro. These commands are only effective when PDFSealer is installed. [DestSignAPName, DestSignCertName, DestSignEnable, DestSignPageNum, DestSignRotateAP, DestSignShowAP] Since Ver 6.09, users can set these options in the UI directly. It requires PDFSealer feature and proper toolbox or Ent. Pack license installed.

The simplest case is just to enable digitally signing. In this case, it will use the default signature appearance settings and certificate to sign the PDF created.

```
@@DestSignEnable: True@@      // Enable Signing PDF by PDFSealer
@@DestSignShowAP: True@@      // Enable showing signature appearance

@@DestSignCertName: Certname@@
// Specify the certificate name; leaving empty will use the default

@@DestSignAPName: demo3@@
// Specify the appearance name; leaving empty will use the default

@@DestSignPageNum: 1@@
// -1 mean the last page
// Page number starts from 1. –1 means the last page.
// –2 means all pages. –3 mean odd pages. –4 mean even pages.
// –2 through –4 are available since version 6.08 of PDFSealer

@@ DestSignRotateAP: 0@@
//0→0 degree; 1→90 degree; 2→180 degree; 3→270 degree
```

Watermark Sample:

Since Ver 6.24 Build 4398, users can control WatermarkEnable and WatermarkSelected by embedded commands directly without changing the registry.

```
@@WatermarkEnable: True@@     // Enable stamping watermark
@@WatermarkSelected: WatermarkName@@ // select watermark defined
```

The “WatermarkSelected” must be assigned to a value, which is already defined in Watermark property pages. If users need to dynamically define or change watermark attribute, users have to use the advanced commands for watermarks.

In addition to per-user watermark in HKCU, administrators can define watermarks in the profile settings in HKLM to allow all users to view and select predefined watermarks. The watermarks defined in HKLM profile will automatically be in the watermark list with a prefix “Global\” in the all user accounts in the machine. For example:

```
@@WatermarkEnable: True@@     // Enable stamping watermark
@@WatermarkSelected: Global\GlobalName@@ // select watermark in HKLM
```

In addition to the global watermarks, administrators may force stamping a global watermark for all PDF created if WatermarkEnable is True in the profile settings in HKLM. In such case, the WatermarkEnable and WatermarkSelected defined in the profile setting in HKLM represents the global mandatory watermark. These settings are separated from the commands or registry settings in HKCU. Users can also stamp extra watermarks no matter the mandatory watermark is enabled or not.

JobSplitPDF

Since Ver 6.10 build 3390, a new command JobSplitPDF is added for forcing create a new PDF since next page. That is split a single PDF into several ones depending on the location page of JobSplitPDF. Since Ver 6.20, a new command “DestSplitJob” is available for simplifying usage of JobSplitPDF. “DestSplitJob” supports the fixed interval splitting to allow assigning the command once.

With this command, developers can easily doing batching processing based on a single print job, e.g., a single query of database with generating for 1000 users in a print job but forcing to split into 1000 PDF files and send to the correct corresponding user.

As explained, JobSplitPDF will force to create a new PDF since next page. Hence multiple JobSplitPDF in the same page does the same effect as one. JobSplitPDF is not a page-break. Instead it is command to tell the driver to save into a new PDF starting next page. It replies on page break from the application or report designer level. For example,

```
@@JobSplitPDF: True@@         // JobSplitPDF
```

Users may also need to specify Filepath correctly for individual PDF when doing split. When JobSplitPDF takes effect, it will create a new PDF for the coming pages. However, it does NOT reset embedded commands or registry settings applied by previous pages. Hence users can easily put the common commands in the very beginning for starting the print job. On every JobSplitPDF, only override those user or PDF depending settings again.

For example,

```
@@CommonSettings: ---@@
.. .. ..

Individual settings
@@EmailTo: $UserEmailAddr@@
@@EmailSubject: Monthly Report for $UserName$ @@

@@JobSplitPDF: True@@         // JobSplitPDF
<- real page break of application report ->


Individual settings
@@EmailTo: $UserEmailAddr@@
@@EmailSubject: Monthly Report for $UserName$ @@

@@JobSplitPDF: True@@         // JobSplitPDF
<- real page break of application report ->


Individual settings

@@JobSplitPDF: True@@         // JobSplitPDF
```

```
<- real page break of application report ->
```

Since Ver 6.20, a new command “DestSplitJob” is available for simplifying usage of JobSplitPDF. “DestSplitJob” supports the fixed interval splitting to allow assigning the command once.

For example, add the command in top of the document or report

```
@@DestSplitJob: 1@@
.. .. ..
```

If commands detected, it will split on every page when converting to PDF. All post-actions including emailing, signing, or overlaying, will apply on every split job. Users can assign the page interval for splitting in this command.

If the report or document style does not have fixed number of pages to split, users have to use JobSplitPDF instead. JobSplitPDF is suitable for report designer without knowing number of pages in advance.

Profile Update:

As said, directly embedded commands can override settings of the current print job for settings in “Email”, “Security”, “Doc Info”, and “Destination”. To provide more flexibility, eDocPrinter PDF Pro also supports “Profile Update” mode by embedded commands. In this mode, the settings will be flushed into registry. And these changes will not be effective until next printing job. This mode can be used with the direct command mode. Users can utilize this feature to update the profile by sending a printing job first. After the update is done, it then sends printing job with the settings specified by these settings.

```
Please change “@@” to “%%” to make these commands effective
“Profile Update” mode must be bounded by 2 special commands
REGProfileSubkey and REGProfileFlush. Only commands in this section
will be flushed into profile registry. Every section can only be used
to update the settings for the category specified by REGProfileSubkey.

@@REGProfileSubkey: Page Setup@@
@@LayoutNup: 6@@
@@LayoutNupBorder: 1@@
@@REGProfileFlush: True@@


@@REGProfileSubkey: Watermark@@
@@WatermarkEnable: True@@
@@WatermarkSelected: Untitled@@
@@REGProfileFlush: True@@
```

Advanced Commands: (ADV)

Command2Action Addin Commands:

(Requires Adv. Pack Licenses)

[Command2Action Addin]: Advanced embedded commands set for supporting adding images, link action, bookmarks, named destinations, and etc. It requires additional Adv. Toolbox license or Adv. Pack License. [%%acbk], [%%acln], [%%acim] are available now for adding bookmarks, links, and images. It's suitable for Database, or ERP report like applications.

Syntax:

acbk: bookmark name[,[mode],[page num],[node cout],[destination name]]

where:

(1) mode 0 means creating bookmark and named destination both; mode 1 means creating named destination only; mode 2 means creating bookmark only
(2) if page number is assigned, it means it will use the page number as destination instead of automatically using the position the [%%acbk] locates.
(3) Node count is for constructing child level of bookmarks. When it is not zero, it means the next coming count of bookmark will be the child nodes of the current one.
(4) If [destination name] is specified, it will be used as the name of the named destination. It's useful when users want different display name (bookmark name) and destination name.

acln: link address[,width,height]

where:

(1) link address can be a URL, a local file representation like C:\demo1.jpg, or an internal cross reference to a destination name or page number. Internal cross reference is represented as #bookmarkname or #page=?
(2) Width and height can be specified for the link rectangle size. Otherwise, it will automatically use the whole command string size as the link rectangle size.
(3) The units for width and height are "points", i.e., 1/72 inch

acim: image path[,width,height]

where:

(1) image path must be a file path for a BMP or JPG image file like C:\demo1.jpg
(2) Width and height can be specified for the image size. It will automatically place the image aligned with the command string start position with image size width and height specified. When height is missing, it will automatically use calculate the height from the width with keeping the aspect ratio. If both are missing, it will

automatically use the image attribute width and height to calculate the size based on 72 DPI. (i.e every pixel representing a point)

(3) The units for width and height are “points”, i.e., 1/72 inch

acef: filename, [icon style[,R, G, B]]

where:

(1) The first parameter is the full path of the file to be embedded.

(2) Users can set the icon style by the optional parameter. Currently, predefined style s includes “Graph”, “PushPin”, “Paperclip”, and “Tag”

(3) Users can set the color in RGB. (each color is from 0 ~ 255)

(4) [acef] is available since ver. 6.18

acnt: comments[,bOpen,R,G,B,[title,type,flag,orgx,orgy]]

where:

(1) The first parameter is the comment text. If the "comments" contain comma ',', it need to add a backslash to escape.

(2) The “type” may be Comment, Key, Note, Help, NewParagraph, Paragraph, or Insert.

(3) [acnt] is available since Ver 6.38.

acpf: the existing PDF path

where:

(1) This special command is used for processing existing PDF assigned instead of the current PDF. Current job file can be used as for pure embedded commands processing. For example, users may want to automatically stamp watermarks on existing PDFs. They may use this command to automate the process by embedded commands with using the [acpf] to specify the existing PDF file.

(2) [acpf] is available since Ver. 6.38.

Examples:

```
Bookmarks:
%%acbk: HereIsABookmark%%
// will create a bookmark and named destination on this position
%%acbk: FirstPageBookmark,0,1%%
// will create a bookmark jumping to page 1
%%acbk: NamedDestONLY,1%%
// will create a named destination only in name “NamedDestONLY”


%%acln: http://www.iteksoft.com/ %%
// create a link in this command string rectangle to URL specified
%%acln: #page=3%%
// create a link jumping to page 3
%%acln: #NamedDestONLY%%
// create a link jumping to the named destination or bookmark created
```

```
%%acln: http://www.google.com,256,256%%
%%acim: c:\demo1.jpg,256,256%%
// place an image here with width and height
// link in rectangle size 256,256 will also be created by acln command


%%acef: c:\logo2006.jpg,Paperclip,0,255,0%%
// embed a file attachment into PDF
// Color is green


%%acnt: My suggestions\, comments\, and opinions, true%%
// add a comment with opening it
// ver 6.38 or later


%%acnt: My suggestions\, comments,false,255,255,0,Title,Key%%
// add a comment and specify color, title, and type attributes
// ver 6.38 or later


%%acpf: c:\temp\existingpdf1.pdf%%
// process the existing PDF with commands in current job
// ver 6.38 or later
```

Command2Action- Watermark Commands:

(Requires Adv. Pack Licenses)

[Command2Action- Watermark Commands]: An advanced embedded commands set for supporting dynamically changing watermark attributes or defining watermarks which are effective in current print job only.

aciw:

This command is used for stamping predefined watermark allowing modifying attributes.

Syntax:

aciw: watermark name, [page range, bBackground, bShowWhenDisplay, bShowWhenPrint]

where:

(1) "Watermark name" is the predefined watermarks in HKCU or HKLM
(2) Users may use this command to stamp watermark with attributes as parameters.
(3) Users may use this command to stamp multiple watermarks
(4) "Page range" represents the pages to be stamped. Special values
    -1: means the last page
    -2: means all pages
    -3: means odd pages
    -4: means even pages
    it can also be simple page number or combination of pages e.g., 3-10 or 1;9;-1
(5) "bBackground" is "true" or "false"
(6) "bShowWhenDisplay" is "true" or "false"; (it requires reader 6.0 or later)
(7) "bShowWhenPrint" is "true" or "false"; (it requires reader 6.0 or later)
(8) [aciw] is available since Ver. 6.24

Example:

```
%%aciw: water1,-3%%          // stamp water1 on odd pages
%%aciw: water2,5-7;-1%%      // stamp water2 on page 5-7 and last page
```

acit:

This command is used for stamping text watermark with all attributes dynamically defined by parameters.

Syntax:

acit: text string, fontsize, R,G,B, [fontfamily, style, placemode, postype, xoff, yoff, bBackground, bShowWhenDisplay, bShowWhenPrint, pagerange, textmode, rotate, opacity, bBorder, borderSize, borderStyle, R, G, B, URI, posUnit]

where:

(1) "text string" is the text to be stamped as text watermarks.

(2) “Fontsize” represents the font size in points.
(3) “R,G,B” are color values from 0~255, e.g., 255,0,0
(4) “Fontfamily” represents the font family name, e.g., Arial.
(5) “Style” represents the font style. 0 is Regular; 1 is Bold; 2 is Italic; 3 is Bold-Italic.
(6) “Placemode” represents the placing method of the text watermark as the list sequence in text watermark editing UI. For example, 0 means placing as “diagonally from LB to RT with sizing automatically”
(7) “Postype” represents the relative position when “placemode” is 3 – Manually. For example, postype 0 means relative to the center of the page.
(8) “Xoff” and “Yoff” represents the x-dir and y-dir offset from the relative position in unit as “PosUnit” (default is points).
(9) “bBackground” is “true” or “false”
(10) “bShowWhenDisplay” is “true” or “false”; (it requires reader 6.0 or later)
(11) “bShowWhenPrint” is “true” or “false”; (it requires reader 6.0 or later)
(12) “Page range” represents the pages to be stamped. Special values
-1: means the last page
-2: means all pages
-3: means odd pages
-4: means even pages
it can also be simple page number or combination of pages e.g., 3-10 or 1;9;-1
(13) “Textmode” represents the drawing mode as the UI. 0 means “Normal”; 1 means “Outline only”; 2 means “Invisble”.
(14) “Rotate” is the rotate in degree of the text watermark.
(15) “Opacity” is the opacity level from 0 to 100. Default value is 100.
(16) “bBorder” is “true” or “false” for drawing border around the text watermark.
(17) “BorderSize” is the border line width in points
(18) “BorderStyle” is either 0 : “Solid” or 1 : “Dashed” as the UI options
(19) “R,G,B” are color values from 0~255, e.g., 255,0,0 for the border line color.
(20) “URI” represents the URL address when clicking the border rectangle.
(21) “PosUnit” is either “points”, “mm” or “inches”; default is “points”
(22) [acit] is available since Ver. 6.24

Example:

```
// example1, stamp text on the current page
%%acit: text stamped,48,255,0,0%%


// example2, stamp text diagonally on odd pages
%%acit_: text stamped2,48,255,0,0,%%
%%acit_: Tahoma,3,0,0,0,0,%%
%%acit: false,true,true,-3%%

// %%acit_ are used for separating commands into multiple lines;
// all advanced commands can utilize this style, e.g. %%aciw_.
```

```
// example3, stamp page number on all pages center-bottom
%%acit_: Page #p1,32,0,0,255,%%
%%acit_: Tahoma,3,3,7,0,0,%%
%%acit_: false,true,true,-2,0,0,100,%%
%%acit: true,2,0,255,0,0,www.google.com,%%

// where #p1 represents the variable of current page number
// #p0 can also be used for representing the total pages
// Please refer command2action variables for details about variables.
```

acim:

This command is used for stamping image watermark with all attributes dynamically defined by parameters.

Syntax:

acim: img path,w,h [, placemode, postype, xoff, yoff, bBackground, bShowWhenDisplay, bShowWhenPrint, pagerange, bWholePage, rotate, opacity, bBorder, borderSize, borderStyle, R,G,B, URI, imgUnit, posUnit]

where:

(1) “img path” is the full path to the jpeg or bmp image.
(2) “w” and “h” represent the width and height. Default “ImgUnit” is “points”. If “h” is omitted, the height is calculated with the original ratio.
(3) “Placemode” represents the placing method of the image watermark in the UI. For example, 0 means placing as “Center automatically”; 1 means “relative to the postype”.
(4) “Postype” represents the relative position when “placemode” is 1 – Manually. For example, postype 0 means relative to the center of the page; 7 means relative to the center-bottom of the page.
(5) “Xoff” and “Yoff” represents the x-dir and y-dir offset from the relative position in unit as “PosUnit” (default is points).
(6) “bBackground” is “true” or “false”
(7) “bShowWhenDisplay” is “true” or “false”; (it requires reader 6.0 or later)
(8) “bShowWhenPrint” is “true” or “false”; (it requires reader 6.0 or later)
(9) “Page range” represents the pages to be stamped. Special values
    -1: means the last page
    -2: means all pages
    -3: means odd pages
    -4: means even pages
    it can also be simple page number or combination of pages e.g., 3-10 or 1;9;-1
(10) “bWholePage” is “true” or “false” representing to fill the whole page or not.
(11) “Rotate” is the rotate in degree of the text watermark.
(12) “Opacity” is the opacity level from 0 to 100. Default value is 100.
(13) “bBorder” is “true” or “false” for drawing border around the text watermark.
(14) “BorderSize” is the border line width in points
(15) “BorderStyle” is either 0 : “Solid” or 1 : “Dashed” as the UI options
(16) “R,G,B” are color values from 0~255, e.g., 255,0,0 for the border line color.

(17) “URI” represents the URL address when clicking the border rectangle.
(18) “ImgUnit” is either “points”, “mm” or “inches”; default is “points
(19) “PosUnit” is either “points”, “mm” or “inches”; default is “points”
(20) [acim] is available since Ver. 6.24

Example:

```
// example1, stamp an image on all pages center
%%acim_: c:\logo.jpg,144,,0,0,0,0,%%
%%acim_: false,true,true,-2,false,0,100,%%
%%acim: true,0,0,255,0,0,www.google.com,%%
```

Command2Action- Variables:

(Requires Adv. Pack Licenses)

[Command2Action- Variables] support resolving environment variables and specially defined date and time variables including %#DATE%, %#TIME%, %#DATEX%, %#TIMEX%%, %#YYYY%, %#MM%, %#DD%, %#HOUR%, %#MIN%, %#SEC%) in settings and embedded commands including DestDir, DestFile, Filepath, EmailSubject, EmailContent, and text watermark field. In addition, %#FNAME%, %#FTPServer% and %#FTPServerDir% variables are available only in EmailSubject and EmailContent command. Text watermark commands and settings also supports 2 special page number variables (%#p1% and %#p0%) representing the current page number and total number of pages.

It requires Adv. Pack licenses for using command2action variables. When using predefined date, time, and page number variables with unregistered adv. pack, it will insert a trial string in the resolved data.

The variables %#DATE% %#TIME% are resolved using Windows regional locale format. It may include special separators, which are illegal characters in the filename. Users may choose %#DATEX% and %#TIMEX%% for resolving without format separator. For example, instead of resolving as “2007-09-18”, the variable %#DATEX% generates “20070918”.

Example:

```
// Using variables in Filepath command
@@Filepath: c:\out\%USERNAME%-%#DATEX%-%#TIMEX%.pdf@@

// Using variables in EmailSubject command
@@EmailSubject: This is created in %#DATE% for %USERNAME% @@

// Using variables in EmailContent command
@@EmailContent: Download from %#FTPServer%\FTPServerDir\%#FNAME% @@
```

Command2Action- Form Commands:

(Requires Adv. Pack Licenses)

[Command2Action- Form Commands]: An advanced embedded commands set for supporting adding form fields into PDF. %%acft, %%acfb, %%acfk, %%acfr, %%acfl, and %%acfs for adding text fields, button, checkbox, radio group, list box, and empty signature form fields.

acft:

This command is used for adding PDF form text field.

Syntax:

acft: fieldname, default text[, fontsize, width, lines, maxchars, bBorder, R,G,B, bkR,bkG,bkB]

where:

(1) "Fieldname" is the internal PDF field name.
(2) "Default text" is the default value of the text field.
(3) "Fontsize" is the field font size.
(4) "Width" is the field width in points.
(5) "Lines" is the number of lines of the text field. Value greater than one means a multi-line text field.
(6) "Maxchars" means the maximum allowable characters in the text field.
(7) "bBorder" is "true" or "false" for drawing border or not.
(8) "R,G,B" are color values from 0~255, e.g., 255,0,0 for the border line color
(9) "bkR,bkG,bkB" are color values from 0~255, e.g., 255,0,0 for representing the background color of the field.

Example:

```
// Text field 1
%%acft: text1,default text,16,,1,40,true,255,0,0,200,200,200%%


// Text field 2, 3 lines with max chars limit
%%acft: text2,Please enter your addr,16,288,3,,true%%
```

acfk:

This command is used for adding PDF form checkbox field.

Syntax:

acfk: fieldname,value,[fontsize,bBorder,R,G,B,bkR,bkG,bkB]

where:

(1) “Fieldname” is the internal PDF field name.
(2) “Value” is the value of the Checkbox, “Yes” means “checked”.
(3) “Fontsize” is the field font size.
(4) “bBorder” is “true” or “false” for drawing border or not.
(5) “R,G,B” are color values from 0~255, e.g., 255,0,0 for the border line color
(6) bkR,bkG,bkB” are color values from 0~255, e.g., 255,0,0 for representing the background color of the field.

Example:

```
// Check box field with value Yes
%%acfk: check1,Yes,,true,255,%%
```

acfr:

This command is used for adding PDF radio button field.

Syntax:

acfr: fieldname,value,selected,[fontsize,bBorder,R,G,B,bkR,bkG,bkB]

where:

(1) “Fieldname” is the radio group field name.
(2) “Value” is the value of the Radio button when selected.
(3) “Fontsize” is the field font size.
(4) “bBorder” is “true” or “false” for drawing border or not.
(5) “R,G,B” are color values from 0~255, e.g., 255,0,0 for the border line color
(6) bkR,bkG,bkB” are color values from 0~255, e.g., 255,0,0 for representing the background color of the field.

Example:

```
// Radio group field with “second” ON by default
%%acfr: radio1,first,second,,true%%

%%acfr: radio1,second,second,,true%%

%%acfr: radio1,third,second,,true%%

%%acfr: radio1,fourth,second,,true%%
```

acfl:
This command is used for adding PDF form listbox field.

Syntax:
acfl: fieldname, value, valuelist, flags, [fontsize, width, bBorder, R,G,B, bkR,bkG,bkB]

where:

(1) “Fieldname” is the listbox field name.
(2) “Value” is the default selected.
(3) “Valuelist” is the items in the listbox, e.g., (item1)(item2)(item3)(item4)
(4) “Flags”is the internal flag value of PDF listbox field.
(5) “Fontsize” is the field font size.
(6) “Width” is the field width in points.
(7) “bBorder” is “true” or “false” for drawing border or not.
(8) “R,G,B” are color values from 0~255, e.g., 255,0,0 for the border line color
(9) “bkR,bkG,bkB” are color values from 0~255, e.g., 255,0,0 for representing the background color of the field.

Example:

```
// List box
%%acfl:list1,item4,(item1)(item2)(item4)(item7),,,72,true%%
```

acfs:
This command is used for adding PDF form empty signature field.

Syntax:
acfs: fieldname,[width,height,bBorder,R,G,B,bkR,bkG,bkB]

where:

(1) “Fieldname” is the listbox field name.
(2) “Width” and “Height” is the field width and height in points
(3) “bBorder” is “true” or “false” for drawing border or not.
(4) “R,G,B” are color values from 0~255, e.g., 255,0,0 for the border line color
(5) “bkR,bkG,bkB” are color values from 0~255, e.g., 255,0,0 for representing the background color of the field.:

Example:

```
// Empty signature field
%%acfs: sig1,144,72,true,0,0,255%%
```

acfb:

This command is used for adding PDF form empty signature field.

Syntax:

acfb: caption,action,url,flags,[fontsize,width,bBorder,R,G,B,bkR,bkG,bkB]

where:

(1) "Caption" is the caption on the button field.
(2) "Action" is the action when clicking the button. Currently, it has to be "SubmitForm", or "ResetForm".
(3) "URL" is the parameter for the action. For example, the URL for submitting the form.
(4) "Flags" is the flags of the action. For "SubmitForm", it controls the submit format and method. For example, value "4" means submitting form by HTML POST. Please refer the PDF specification for details.
(5) "Fontsize" is the field font size.
(6) "Width" is the field width in points.
(7) "bBorder" is "true" or "false" for drawing border or not.
(8) "R,G,B" are color values from 0~255, e.g., 255,0,0 for the border line color
(9) "bkR,bkG,bkB" are color values from 0~255, e.g., 255,0,0 for representing the background color of the field.:

Example:

```
// Submit button
%%acfb: SUBMIT,SubmitForm,http://yoururl/demoform1.php,4,16,144,true%%
```

Redirect (Redirect2Print Addin):

(Requires Adv. Pack Licenses)

Redirect2Print Addin -- Support redirecting current print job to other printers simultaneously. Since 6.16, users can enable redirecting current print job to other printers when creating the PDF. Users have to enable “Enable redirecting current print job to:” I n the [Redirect] property page. This option cannot be controlled by embedded commands on the fly. It must be set by registry or UI in the profile settings before starting print job just as enabling embedded commands detection.

When enabled, users can set the printer list by embedded commands. When the printer list is empty, the print job will not be redirected.

```
Please change “@@” to “%%” to make these commands effective. Assume
Redirect2Print is enabled.

@@RedirectApplyFormat: True@@
// This option will enable applying overlay/merge/watermark on
redirecting. If it is false, it will redirect the original printing
content to other printers only.
@@RedirectSaveSPL: True@@
// This option will store the print job in (*.pdj) to enable support
overlay or merge in future redirecting.

@@RedirectPrinterList: (Apple LaserWriter)(HP LaserJet)@@
```

This command will add 2 printers “Apple LaserWriter” and “HP LaserJet” for redirecting. Every printer in printer list is represented as a string by parenthesis. It can be a simple string representing the printer name only. Or it can be a string containing other parameters separate by comma. If other parameters are not specified, default values will be used. For example:

```
@@RedirectPrinterList: (Apple LaserWriter,False,False,0,False,False,,False,,1,False,)@@
// parameter 2: True/False: Show preference dialog before printing: Default False
// parameter 3: True/False: Select printer paper size automatically: Default False
// parameter 4: 0,1,2: Page scaling mode: Default 0 None scaling
// parameter 5: True/False: Auto-rotate: Default False
// parameter 6: True/False: Auto-center: Default False
// parameter 7: Bin name: Paper source bin name: Default
// parameter 8: True/False: Printer to File: Default False
// parameter 9: File path: File path if Print to File is enabled
// parameter 10: 1,2,3…: Copies: Default 1
// parameter 11: True/False: Collate: Default False
// parameter 12: image DPI; for ->JPG and ->BMP; available since 6.18
// parameter 13: B/W or Color printing; available since 6.24 for printer supporting that
```

Be careful about the command line length. Each command must be in a single line. If it is too longer to fit into one line, users have to separate it into multiple lines with using special symbol “:” to concatenate these commands. For example:

```
@@RedirectPrinterList: (Apple LaserWriter)@@
```

```
@@RedirectPrinterList: :(HP LaserJet)@@
```

If the printer name contains back slash, it needs to enter the back slash twice for escape. For example, to redirect to a server printer \\Server1\HPLaserJet

```
@@RedirectPrinterList: (\\\\Server1\\HPLaserJet)@@
```

Since Ver 6.18, there are 2 special printers “->JPG” and “->BMP” available. These two are for Redirect2JPG and Redirect2BMP for the current print job. It will use the Filepath and the auto-number format to generate the image names with proper file extension (.JPG or .BMP). The auto-numbered format is controlled by registry “DestAutoNumFormat” in “Destination” category. Its default value is “-%03d.pdf”.

Since Ver 6.24, it adds a new Color option for selecting printing as B/W or Color mode for Color printers supporting this parameter.

Since Ver 6.34, special printers “->TIF” and “->TIFG3” are added for redirecting as color or black/white TIF images. It also adds a special "->DEF" instance for supporting redirect to the default printer automatically. The Windows has default printer as per-user settings.

Since Ver 6.34, when doing overlay with template images (bmp, jpg, tiff) directly, Redirect2Print does not require creating template PDJ files in advance.

Since Ver 6.38, the optional "Use Reader to Printer" is available for launching the PDF reader to print without the need of pdj. It requires reader 6 or later. The command use is like: `@@RedirectUseReader: True@@`

Commands of Ent. Pack Features: (ENT)

FTP Commands:

(Requires Ent. Pack Licenses)

It supports uploading the result PDF by FTP. The FTP settings can be controlled by UI options (registry settings) and embedded commands including FTPEnable, FTPPassive, FTPServer, FTPPortNum, FTPServerDir, FTPUserName, and FTPShowProgress.

PDFSealer Commands:

(Requires Ent. Pack Licenses)

There are commands (registry settings) for controlling calling PDFSealer silently to digitally sign PDF created. These commands are only effective when PDFSealer feature is installed. In Destination property page, users can select the "Signing by PDFSealer" option for proper settings. The commands available include DestSignAPName, DestSignCertName, DestSignEnable, DestSignPageNum, DestSignRotateAP, and DestSignShowAP.

The simplest case is just to enable digitally signing. In this case, it will use the default signature appearance settings and certificate to sign the PDF created.

```
@@DestSignEnable: True@@      // Enable Signing PDF by PDFSealer
@@DestSignShowAP: True@@      // Enable showing signature appearance

@@DestSignCertName: Certname@@
// Specify the certificate name; leaving empty will use the default

@@DestSignAPName: demo3@@
// Specify the appearance name; leaving empty will use the default

@@DestSignPageNum: 1@@
// -1 mean the last page
// Page number starts from 1. –1 means the last page.
// –2 means all pages. –3 mean odd pages. –4 mean even pages.

@@ DestSignRotateAP: 0@@
//0→0 degree; 1→90 degree; 2→180 degree; 3→270 degree
```

Registry settings:

By default, eDocPrinter PDF Pro supports multiple printer profiles. Each printer profile has its own per-user settings stored in HKEY_CURRENT_USER and global setting in HKEY_LOCAL_MACHINE. The settings are in

HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\[Profile Name]
HKEY_LOCAL_MACHINE\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\\[Profile Name]

where the [Profile Name] is the printer instance name. For example, the printer "eDocPrinter PDF Pro" has its per-user settings in

HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter PDF Pro

For office-addin settings, there is no profiles. The settings are per-user (HKCU) and HKLM.

HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Word Addon
HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Excel Addon
HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\PowerPoint Addon

The 5.0 is a fixed entry. It is not related to the version of eDocPrinter.

Special Registry Settings:

Global Settings

Every eDocPrinter printer instance has its own per-user profile setting in HKEY_CURRENT_USER and its per-machine global setting in HKEY_LOCAL_MACHINE.

FlagProhibitHKCU

The registry key "FlagProhibitHKCU" is special key-value pair for forcing the sub-category settings to be read only from HKLM. For example, by setting the special flag "FlagProhibitHKCU" to "True" in

HKEY_LOCAL_MACHINE \Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter PDF Pro\Security

it will force driver to read only the settings from the same key in HKLM only.
Since 6.16 bulid 3902, it already support MUST key for per key-value mandatory settings, we do not recommend using FlagProhibitHKCU. Please use MUST profile settings instead.

FlagProhibitShowPage

The registry key “FlagProhibitShowPage” is special key-value pair for hiding the sub-category property page in the UI. For example, by setting the special flag “FlagProhibitShowPage” to “True” in

HKEY_LOCAL_MACHINE \Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter PDF Pro\Security

it will hide the property page of Security. Administrators can use PropAdm to set this value too.

PropAdm

“PropAdm” is a simple Windows utlity providing a simple user interface for Administrators to control the global settings of eDocPrinter PDF Pro for multi-user environment in HKEY_LOCAL_MACHINE (a.k.a. HKLM). Hence users have to own Administrator permission to run this utility to changing settings in HKLM.

This utility provides showing property pages with settings in HKLM instead of the HKCU. Hence administrators can assign the default settings in HKLM in UI directly without using editing the registry directly.

By the proprietary flag of FlagProhibitHKCU, administrators can force all users must use the settings from HKEY_LOCAL_MACHINE and cannot override them from the user interface by their own settings in HKEY_CURRENT_USER. Since Ver 6.16, we recommend administrators using “MUST” profile settings for per key-value enforcement instead of “FlagProhibitHKCU”. Currently, administrators have to manually editing the registry for adding using “MUST” profile and its per-key value settings.

For the special Watermark category, the settings in HKEY_LOCAL_MACHINE are for mandatory settings. That means administrators can force the global watermark be printed to all PDF documents created from all users in the machine. For example, administrators can force all PDF documents printed with a company logo with a hyperlink to company website.

After installation, “PropAdm” will be installed in the same directory as eDocPrinter PDF Pro by default. There is a shortcut in the “eDocPrinter PDF Pro” group in “Windows -> Start -> Programs”.

MUST profile settings

Since ver 6.16 build 3902, it supports MUST key for storing mandatory settings. This is prefixed before profile name, for example,

HKEY_LOCAL_MACHINE \Software\ITEKSOFT\eDocPrinter\PDF\5.0\MUST\Profiles\eDocPrinter PDF Pro

This method is different from the “FlagProhibitHKCU”, which disable reading HKCU for the whole category (subkey), e.g., “Security” or “Email”. By “MUST” approach, administrators can set mandatory profile setting in per key-value based.

Administrator can just force a sepecific entry to be False instead of disabling reading the whole sub-category settings in HKCU. As before, the profile settings in HKLM is the global default settings. The profile with prefixed \MUST\ in HKLM will be the mandatory settings. By default, it will not affect old settings since there is NO MUST entries set in HKLM.
For example, administrator adds the 2 key-value pairs in MUST registry entry

HKEY_LOCAL_MACHINE \Software\ITEKSOFT\eDocPrinter\PDF\5.0\MUST\Profiles\eDocPrinter PDF Pro\Security

EnableOwnerPasswd True
OwnerPasswd ***

It will force all PDF created in this machine will be encrypted with administrator assigned owner password.

For example, administrator adds the 1 key-value pairs in MUST registry entry

HKEY_LOCAL_MACHINE \Software\ITEKSOFT\eDocPrinter\PDF\5.0\MUST\Profiles\eDocPrinter PDF Pro\Destination

DestPostSaveDlg True

It will force prompting SaveAs after job spooled for all users in this machine. Administrators can use the similar way to create a group policy by writing proper values into HKLM.

RemoteRegServer setting

Since Ver 6.24 build 4398, a special registry setting “RemoteRegSever” can be set to force reading global default and MUST settings from remote server HKLM instead of local machine HKLM.

Administrators can manually add a string value with name “RemoteRegSever” in the key

HKEY_LOCAL_MACHINE \Software\ITEKSOFT\eDocPrinter\PDF\5.0
RemoteRegServer -> “servername”

Global Watermarks settings

Please refer watermark section for details.

Page Setup

There are registry settings, which can control the margin in left, top, right, and bottom, in HKCU\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter PDF Pro\Page Setup when LayoutDirMarginEnable is True

LayoutLeftMargin
LayoutTopMargin
LayoutRightMargin
LayoutBottomMargin
These represent values of the margin in unit as the unit in Page Setup

These settings now can only be set by registry, (These cannot be controlled by embedded commands as margins are set in the every beginning of printing) Since Ver 6.20, these settings are directly available in UI, hence users can set these in UI directly.

Shell Extension customization settings

Administrator can customize shortcuts in by modifying registry values under HKLM\SOFTWARE\ITEKSOFT\Drag2PDF\ShellExt. By default, there are 3 shortcuts defined, "->PDF", "->PDF&Email", and "->PDF&Merge".

The key name is representing the shortcut title. In addition, "#" can be prefixed to enforce the ordering. For example, "#1->PDF" means the shortcut "->PDF" will be on the first position of the group.

The value name is representing the command line to call the Drag2PDF for conversion. For example, [-sm0 –sp"Drag2PDF"] is the command line for "->PDF". The "-sm" represents the converting mode of Drag2PDF. For example, "-sm0" will do "Convert to PDF files" and "-sm1" will do "Convert and Merge into one PDF". "-sp" is used for specifying the printer profile for converting. By default, it is "Drag2PDF". Administrators may add additional printer instance with different settings mainly for right-click conversion actions of shell extension. "-se" will force to enable Email as in the Email property page. It follows the email settings defined in the printer profile.

PDF/A-1b registry settings

Under [Doc Info] registry, three registry values (DocIntentMode, DocMetaMode, and DocPDFVersion) must be set properly for PDF/A-1b in addition to force embedding all used fonts. When installing properly, these values shall be (all string type registry entries)

"DocMetaMode" "1"
"DocIntentMode" "1"
"DocPDFVersion" "1.4"

Printer Server registry settings

Since Ver. 6.30, eDocPrinter supports a “printer server” mode for accepting printing jobs from other computers, which may not have eDocPrinter installed. It requires intranet or internet server license (per-server).

eDocPrinter PDF Pro is a native Windows printer driver with Office-addin support for creating PDF files on local or network file systems. For fully functioning with support of prompting UI and Office addin interactions, it is required to install eDocPrinter PDF Pro driver on necessary machines locally.

The new printer server mode is designed for special purpose usage, which will do the printing and post-processing in the server. However, under this server mode,

It does NOT support prompting SaveAs dialog or similar UI dialogs.
It does NOT support Office-addins for internal links, cross references, or headings->bookmarks conversions.
It does NOT support launching PDF readers or Email clients in clients.
It does NOT support Drag2PDF.

As a result, it may not be suitable for ordinary customers.

Administrator have to add or set extra printer instances by “PropAdm” or “Add New Printer” shortcut to enable printer server mode in addition to sharing the printer instance by printer control panel. These actions will enable “print server” mode for the eDocPrinter instances (profiles).

Internally, these steps will attach the “eDocPDF” printer processor and add a registry PrinterServerMode under HKLM\SOFTWARE\ITEKSOFT\eDocPrinter\PDF\5.0\ Profiles\“PrinterName”, where “PrinterName” is the printer profile name. Currently, mode “1” is available for PrinterSeverMode. Mode “1” representing the printer instance will work under printer server mode. Printer server mode does not support prompting UI (e.g. SaveAs dialog) because the printing jobs are processed all in the server side. By default, it automatically generates the filename and saves the PDF into the user’s My Documents folder. And it does not launch PDF reader. Usually the user’s context is determined by the logon session or printer connection session in the active directory. Normally, it is the logon user context if printer and the client are both in the same active directory domain.

Basically, printer server mode provides a function similar to the network sharing or active directory printers without installing eDocPrinter drivers on client workstations. Users may change the profile settings, e.g., to enable detecting embedded commands, stamping watermarks, or etc. in the user’s profile in the server. Multiple printer servers are supported in the same way as multiple printer instances of eDocPrinter PDF Pro.

ExtEscape Control:

If developers have the Printer DC (DeviceContext) directly in their own applications, developers can use Windows standard API ExtEscape to control eDocPrinter PDF Pro directly without writing registry.

Basically, it is similar to control by registry settings or commands, but using a specific API to send the key-value pair to the printer DC directly. As it is sent to printer DC, the settings are applicable to the current print job. When writing into registry directly, developers have to restore its original settings to avoid unwanted behaviour after printing.

For details and samples, please refer to the section of Escape Codes and Samples in Appendixes. In addition to the code snippets, users can download one full source example [ESCDEMO], which contains a working example of controlling eDocPrinter PDF Pro programmatically in a simple text editor inherited from MFC “CEditView”.

Synchronization

Synchronization in Application Level:

If applications need to be signaled when PDF creation and post-processing finished, please refer the following methods there are several

I. Executing Post Action Command line: (available since Ver 5.18)

Users can select the "DestActionMode" from the UI to execute a command line after PDF fully created. Using this simple way, users may provide proprietary scripts or EXE file to be executed and do the necessary notification or post-processing in this step before starting the next print job.

Commands, registry settings, and ExtEscape all can control "DestActionMode", "DestCommand", and "DestArgument" at run-time.

II. Polling Sync File Existence: (available since Ver 6.00)

There is a registry setting "DestCreatePDFSync" in HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter PDF Pro\Destination, where "eDocPrinter PDF Pro" is the printer profile (instance) name. If there are multiple eDocPrinter instances, proper instance name shall be used. Its default value is "False". When it is "True", when PDF creation and post-processing (e.g. appending and emailing) are fully finished, a file with the same name as the final PDF but with appending extension ".end" will be created in the same directory of the destination PDF. For example, if the final PDF is "C:\test1.pdf", if creating sync file is "True", it will create a file "C:\test1.pdf.end" as the sync file. Checking the existence of the sync file will know the state of PDF conversion process.

Developers have to make sure the sync file does not exist before print job started and have to delete the sync file after PDF conversion process finished.

Embedded commands and ExtEscape can also control this setting directly.

III. Waiting for Windows Event: (available since Ver 5.52)

Developers can use ExtEscape to set "JobPDFReadyEvent" value to the event name specified. Developers can the use standard Windows API CreateEvent and WaitForSingleObject for creating the event for waiting to be signaled. Developers have to create this event and start waiting for it before print job starting.

Though this “JobPDFReadyEvent” is not belonged to the “Destination” category, developers can also write this key-value into HKEY_CURRENT_USER\Software\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter PDF Pro\Destination for fully controlling eDocPrinter by registry settings. Please make sure to set and clean this key-value before job started and after job finished.

Developers can also use the command “JobPDFReadyEvent” for setting an event name to be signaled when PDF creation and post-processing is finished. For example, adding a command
%%JobPDFReadyEvent: MyPDFCreationISOK%%
in the document to be printed. Developers then create and wait for this event name of “MyPDFCreationISOK” in their applications.

IV. Waiting for Windows Message: (available since Ver 6.19 build 4127)

For message based applications, developers can control the key-value pair “DestWaitPDFReadyMsg” in “Destination” to set the message name. This message will be broadcasted to all applications with message ID got from Windows API “RegisterWindowMessage”. The *wParam* is the print job identifier returned by StartDoc. Its default value is “MsgEDOCPDFPostProcessingOK”.

As the same, this key-value pair can also be controlled by ExtEscape or embedded commands. Just developers have to ensure the version of eDocPrinter for supporting this function.

Please refer to the appendixes for a WORD VBA Macro sample to how to control [Destination] path and wait for PDF finish event.

Synchronization in Driver Level:

Ver 6.18 or earlier:

In Ver 6.18 and earlier, the driver level provides 2 registry settings for the synchronization of PDF creation.

I. "DestLaunchProcess" – True/False ; i.e "Wait PDF if applications closed" in UI option.

   It is available since Ver 5.50. The default value is "True" since Ver 6.14. It will launch a process "epdfact.exe" for waiting PDF until it is fully processed. It is necessary for cases, where applications print and quit after job spooled before PDF is fully ready. (e.g. some batch mode printing)

   For high throughput without blocking, PDF processing work (including, encryption, appending, overlaying, emailing, and etc.) is done in a separate thread. When "DestLauchProcess" is "False", the thread is attached to the application itself. (Hence if the application close, the PDF may not be fully processed) When "DestLauchProcess" is "True", the thread is attached to the "epdfact.exe".

   Since 6.12, there is a registry "DestAutoWaitByProcess" added with default value "True". When it is "True", it will automatically turn on "DestLauchProcess" when it detects the printing is under batch command line printing (i.e. applications may terminate after print job spooled)

   Since 6.14, an extra setting "DestEPDFACTMode" is added for supporting running epdfact.exe as a singleton monitor process. Default value is "1" since Ver 6.14. When "DestEPDFACTMode" is "1", the "epdfact.exe" will run as a singleton monitor process, when first job printed. It will not terminate itself until users log off. When "DestEPDFACTMode" is "0", the "epdfact.exe" will be created and terminated after PDF fully created for every printing job. (Since Ver 6.20, mode "1" is obsolete, please refer the section below for new [DestWaitPDFMode] details).

II. "DestWaitPDFReadyEvent" – True/False; i.e. "Wait synchronization of PDF ready event" (available in registry setting since 6.06, available in UI option since 6.14)

   It ensures the spooler will start printing next job when the previous PDF processing is fully finished. The spooler will not start printing next job until the current PDF processing or merging is fully finished.

Ver 6.20 or later:

Since Ver 6.20, for simplifying and improving the batch processing, a new registry setting "DestWaitPDFMode" is added for controlling the above settings automatically, which value is "0/1/2/3" representing the following methods:

[0] "Process PDF asynchronously"

This is the default mode, which means there is no special driver level synchronization method applied. Users can select to enable or disable "Wait PDF if applications closed" (i.e. "DestLaunchProcess") In this mode, if enabled "DestLaunchProcess", epdfact.exe will be executed and terminated when it finishes the PDF job.

In this mode, it will set the setting values for the current print job as:
DestLaunchProcess -> "As user control"
DestEPDFACTMode -> "0"
DestWaitPDFReadyEvent -> "False"
DestWaitPostAction -> "False"

[1] "Process PDF sequentially"

This mode will queue all PDF processing requests and process one by one sequentially. It guarantees any PDF merging operation will be finished before next PDF processing. It is suitable for common batch appending or printing applications, which need PDFs created sequentially. It provides higher throughput than mode 2 as it allows current job in spooler and PDF merging processing in another thread for previous job simultaneously executed. And it will not the block the spooler printing.

In this mode, it will set the setting values for the current print job as:
DestLaunchProcess -> "True"
DestEPDFACTMode -> "2"
DestWaitPDFReadyEvent -> "False"
DestWaitPostAction -> "As user control"

[2] "Process PDF with full synchronization"

This is similar to "mode 1" above. In addition, it force synchronizing with spooler print jobs and post-action command line execution. (i.e., it guarantees PDF is fully ready when print job status is completed in spooler.) It is mainly for applications, which rely on the Windows API of print job status from the spooler. Otherwise, "mode 1" will be sufficient, since "mode 1" provides higher throughput as it allows current job in spooler and PDF processing of previous jobs simultaneously executed.

In this mode, it will set the setting values for the current print job as:
DestLaunchProcess -> "True"

DestEPDFACTMode -> “2”
DestWaitPDFReadyEvent -> “True”
DestWaitPostAction -> “True”

[3] “Process PDF with spooler synchronization”

This mode is only for backward compatibility, suggest users to select mode 1 or 2. This mode is just the same as enabling only “DestWaitPDFReadyEvent” (i.e. “Wait synchronization of PDF ready event”) in earlier versions.

In this mode, it will set the setting values for the current print job as:
DestLaunchProcess -> “As user control”
DestEPDFACTMode -> “As user control”
DestWaitPDFReadyEvent -> “True”
DestWaitPostAction -> “As user control”

In Ver 6.20, the default value of “DestWaitPDFMode” is “0”, and the default values for others are:

DestLaunchProcess -> “True”
DestAutoWaitByProcess -> “True”
DestEPDFACTMode -> “0”
DestWaitPDFReadyEvent -> “False”
DestWaitPostAction -> “False”

Appendixes

Appendix I: Registry Key-Value tables

Email

Key Name	Value (String)	Purpose/Remark	Availability		
			UI	Command	Since Ver.
EmailEnable	True/False		V	V	
EmailSendMethod	0,1,2	0: Launch Default email client 1: Send silently by default email client 2. Send directly by SMTP 3. Send by Notes Client	V	V	
EmailTo		Addresses	V	V	
EmailCc		Addresses	V	V	
EmailBcc		Addresses	V	V	
EmailFrom			V	V	
EmailSubject		Subject	V	V	
EmailContent		Content	V	V	
EmailContentFile	File path	Content in File		V	
EmailSendAsHTML	True/False		V	V	
EmailUseEMAPI	True/False		V	V	6.12
EmailUseOutlook	True/False		V	V	6.12
EmailMAPIProfile		Profile name		V	
EmailSMTPServer			V	V	
EmailSMTPAuthMode	0,1,2		V	V	
EmailSMTPUserName			V	V	
EmailSMTPPassword			V	V	
EmailSMTPUsingPort	True/False		V	V	
EmailSMTPPortNum			V	V	
EmailAttach	File path		V	V	
EmailBodyEncoding				V	
EmailImportance	0/1/2		V	V	6.18
EmailReturnReceipt	True/False		V	V	6.18
EmailSendNow	True/False	when “Use Outlook”	V	V	6.18
EmailAttachSelf	True/False	Default True		V	6.24
EmailAttachExt		Attach files of the same destination PDF path with the specified extension. For example, .tif		V	6.34

Doc Info

Key Name	Value (String)	Purpose	Remark	Availability		
				UI	Command	Since Ver.
Title				V	V	
Subject				V	V	
Author				V	V	
Keywords				V	V	
DocOpenActionEnable	True/False			V	V	
DocOpenPageNum		Page number		V	V	
DocOpenPageZoom	0,1,2,3,4,5,6,…	Subject		V	V	
DocOpenLayoutMode	0,1,2,3	0: Default		V	V	
DocOpenPageMode	0,1,2,3	0: Deafult		V	V	
DocViewHideMenubar	True/False			V	V	6.14
DocViewHideToolbar	True/False			V	V	6.14
DocViewHideWindowUI	True/False			V	V	6.14
DocViewPrintScaling	0,1	1: Default		V	V	6.14
DocInfoFilenameAsTitle	True/False			V	V	6.16
DocInfoEnableCustom	True/False	Add customized field			V	
CustomDocInfoXXXX		Prefixed with CustomDocInfo			V	
DocPDFVersion	1.3/1.4/1.5/1.6	PDF version header, default value is 1.3				6.30
DocIntentMode	0,1	Add intent info				6.30
DocMetaMode	0,1	Add meta info				6.30
DocViewDisplayDocTitle	True/False	Show Title on Window		V	V	6.34

Security

Key Name	Value (String)	Purpose	Remark	Availability		
				UI	Command	Since Ver.
EnableUserPasswd	True/False			V	V	
EnableOwnerPasswd	True/False			V	V	
UserPasswd				V	V	
OwnerPasswd				V	V	
EncryptLevel	40/128			V	V	
E40_NoPrint	True/False			V	V	
E40_NoEdit	True/False			V	V	
E40_NoCopy	True/False			V	V	
E40_NoComment	True/False			V	V	
E128_Access	True/False			V	V	
E128_Copy	True/False			V	V	
E128_Edit	0,1,2,3,4			V	V	
E128_Print	0,1,2			V	V	

Destination

Key Name	Value (String)	Purpose/Remark	Availability		
			UI	Command	Since Ver.
DestPostSaveDlg	True/False	Users have to set this option in profile (UI) setting directly. It cannot be controlled by commands. Setting to True for changing DestSaveMode and other settings on the fly by embedded commands.	V		6.14
DestSaveMode	0/1/2	0: Prompt SaveAs dialog 1: Save using the name specified by DestFile and DestDir 2: Save using the name auto-generated with DestDir	V	V	
DestDir			V	V	
DestFile			V	V	
DestExistMode	0/1/2/3/4/5	0: Prompt warning dialog 1: Replace directly 2: Auto numbered 3: Merge by Append to 4: Merge by Insert Before 5. Merge by Overlay	V	V	
DestStartNum			V	V	
DestAutoNumFormat		Auto-num format. Default value is “-%03d.pdf”	V	V	
DestEnablePostAction	True/False		V	V	
DestActionMode	0/1/2/3	0: Launch PDF reader 1: Execute command line 2/3: call PDFSealer 4: Remove the final PDF	V	V	
DestCommand		The postaction command line.	V	V	
DestArgument		“%s” represents the final PDF file path	V	V	
DestLaunchProcess	True/False	Wait PDF if application closed	V		5.50
DestAutoWaitByProces	True/False	Automatically turn on DestLaunchProcess when detecting in commandline printing	X	X	6.12
DestEPFACTMode	0/1/2	Mode 0: run epdfact.exe for every print job Mode 1: epdfact.exe is run as a singleton monitor process Mode 1 is obsolete since Ver 6.20. Since 6.20, DestEPDFACTMode is controlled automatically by DestWaitPDFMode	X	X	6.14 6.20
DestWaitPDFReadyEvent	True/False	Wait Synchronization of spooler job finish event Since Ver 6.20, Use DestWaitPDFMode instead	X	V	6.06 6.14 6.20
DestWaitPDFReadyMsg		RegisterWindowMessage for signaling PDF is fully created	X	V	6.20

DestWaitPDFPostAction	True/False	If True, it will wait until the post-action process command line is terminated before ending the current job.	X	V	6.20
DestWaitPDFMode	0/1/2/3	New option for controlling automatically the DestLaunchProcess, DestEPFACTMode, DestWaitPDFReadyEvent, DestWaitPDFPostAction easily. Please refer the section driver level synchronization for details.	V	X	6.20
DestCreatePDFSync	True/False	Create a *.pdf.end when PDF finished	V	V	
DestLogEnable	True/False		V		
DestRemoveFinalPDF	True/False	Remove PDF file after post-processing		V	
DestFastWebView	True/False	Optimized for fast-web view	V	V	
DestMergeUpdate	True/False	Save PDF by incremental update when doing merge.		V	
DestOverlayPDFTemplateEnable	True/False		V	V	
DestOverlayPDFTemplatePath		Template PDF file path	V	V	
DestOverlayType	0/1/2/3	0: As Foreground 1: As Background 2: Appended after each page 3: Insert before each page 2,3 available since Ver 6.20 4: As Foreground and Appended 5: Inserted and As Foreground 6: As Background and Appended 7: Inserted and As Background 4,5,6,7 are available since Ver 6.30	V	V	
DestOverlayWithType	0/1/2/3	As above, available only after Ver 6.20.	V	V	6.20
DestOverlayStretchFit	True/False	Fill the whole page	V	V	
DestOverlayRepeatLast	True/False	Repeat Last Page	V	V	
DestOverlayRepeatInterval	0/1/2/3	Number of template pages to repeat, when RepeatLast is True	V	V	6.20
DestOverlayLastPageIsCover	True/False	Last page is cover	V	V	
DestOverlayBackCoverNum		Number of back-cover pages	V	V	6.20
DestOverlayNewPDFAsStationary	True/False	Newly created PDF as stationary when [Merge by Overlay]	V	V	
DestOverlayPDFTemplateAsStationary	True/False	Template PDF as stationary when [Overlay with]	V	V	
DestOverlayWithAsMergeWith	True/False	Working as [Merge] with	V	V	
DestOverlayPrependNum		Prepend first [number] pages	V	V	
DestOverlayAppendNum		Append last [number] pages	V	V	
DestOverlayRotate		Rotate in degree	V	V	
DestOverlayNewLayerMergeEnable	True/False	When [Merge by Overlay], create new layer	V	V	

DestOverlayNewLayerTemplateEnable	True/False	When [Overlay with], create new layer	V	V	
DestOverlayLayerMultiLevelEnable	True/False	Support multi-level layer name with delimiter '/'	V	V	
DestOverlayMergeLayerName		Layer name for [Merge by Overlay]	V	V	
DestOverlayTemplateLayerName		Layer name for [Overlay with]	V	V	
DestOverlayShowWhenDisplay	True/False	Show the layer when viewing	V	V	6.20
DestOverlayShowWhenPrint	True/False	Show the layer when printing	V	V	6.20
DestSignEnable	True/False	Enable signing PDF created by calling PDFSealer silently	V	V	
DestSignCertStore				V	
DestSignCertName			V	V	
DestSignAPName		Signature appearance name defined in PDFSealer	V	V	
DestSignShowAP	True/False		V	V	
DestSignPageNum		Page number starts from 1. –1 means the last page. –2 means all pages. –3 means odd pages. –4 means even pages. –2 through –4 are available since version 6.08 of PDFSealer	V	V	
DestSignRotateAP	0/90/180/270		V	V	
DestSignFilterMethod	0/1	0: PPKLite 1: PPKMS	V	V	

Watermark

Key Name	Value (String)	Purpose	Remark	Availability		
				UI	Command	Since Ver.
WatermakrEnable	True/False	Enable Watermark	Since Ver 6.24, it can be controlled by commands.	V	V	
WatermarkSelected		Watermark name selected	Since Ver 6.24, it can be controlled by commands	V	V	

Redirect (Redirect2Print Addin)

Key Name	Value (String)	Purpose	Remark	Availability		
				UI	Command	Since Ver.
RedirectEnable	True/False	Enable Redirect2Print	Must be enabled before printing	V		6.16
RedirectPrinterList		Printer List		V	V	6.16
RedirectApplyFormat	True/False			V	V	6.16
RedirectSaveSPL	True/False			V	V	6.16
RedirectUseReader	True/False	Call reader for redirect2print	Need reader 6 or later	V	V	6.38

FTP (FTP Addin, Ent. Pack)

Key Name	Value (String)	Purpose	Remark	Availability		
				UI	Command	Since Ver.
FTPEnable	True/False	Enable FTP		V	V	6.38
FTPPassive	True/False	Passive mod		V	V	6.38
FTPPassword		User password		V	V	6.38
FTPPortNum			Default 21	V	V	6.38
FTPServer		Server name	Need reader 6 or later	V	V	6.38
FTPServerDir		Directory		V	V	6.38
FTPShowProgress	True/False	Show progress		V	V	6.38
FTPUserName		FTP Username		V	V	6.38

Appendix II: Call Office-Addin to Create PDF by VBA

The following VBA sample demonstrates how to call addin to create PDF by the addin from VBA macro.

```
Private Function GetBtnFromTag(cmdbar As CommandBar, tag As String) As CommandBarControl
    Dim item As CommandBarControl

    Set GetBtnFromTag = Nothing

    For Each item In cmdbar.Controls
        If item.tag = tag Then
            Set GetBtnFromTag = item
            Exit For
        End If
    Next
End Function

Sub calleDocPDFAddin()

     Dim stdBar As CommandBar
     Dim ctlbtn As CommandBarControl

     Set stdBar = CommandBars("Standard")

     Set ctlbtn = GetBtnFromTag (stdBar, "edocpdfproButton")

     If Not (ctlbtn Is Nothing) Then
         MsgBox "print to PDF now"
         ctlbtn.Execute
     End If
End Sub
```

For Office 2007, the following VBA sample demonstrates how to call addin to create PDF by the addin from VBA. This method is also applicable to Office 2000/XP/2003 for Ver 6.34 of eDocPrinter or later.

```
Sub calleDocPDFAddin ()

    Dim addin As COMAddIn
    Dim epdfObj As Object

    Set addin = Application.COMAddIns("eDocPDFAddin.WordAddin")
    ' For Excel or Powerpoint, users have to get the proper eDocPDFAddin.ExcelAddin
    ' or eDocPDFAddin.PowerPointAddin

    Set epdfObj = addin.Object
    If Not (epdfObj Is Nothing) Then
        epdfObj.RibbonCommand ("eDocPrinterAddin2PDFBtn")
    End If

End Sub
```

Appendix III: Escape Codes

(available since eDocPrinter PDF Pro Ver 5.51 build 1750 or later)

Job Setting Escape Code Defined:

```
PDFFILENAME                          4312  // Escape Code
DESC_EPDF_PERJOB_KEYVAL_UNICODE      5168  // Escape Code
DESC_EPDF_OTHERS_KEYVAL_UNICODE      5169  // Escape Code
DESC_EPDF_PDFFILENAME_UNICODE        5170  // Escape Code

DESC_EPDF_RESET_ALL_KEYVAL           5180  // Escape Code
```

PDFFILENAME: *compatible with PDF writer API for suppressing save dialog*

DESC_EPDF_RESET_ALL_KEYVAL: *for clearing print job settings by Escape Codes. We suggest developers calling this before all other escape calls.*

DESC_EPDF_PERJOB_KEYVAL_UNICODE: *support to override per-job settings including Security, Email, Doc Info, and Destination as embedded commands.*

DESC_EPDF_OTHERS_KEYVAL_UNICODE: *It is used for override setting values of Page (including N-up, Border, and etc.), Compression, Font Embedding, Links, Watermarks, and Bookmarks. Currently it supports only suppressing common watermark property values, e.g., watermark selected and enabling. It cannot create new watermarks from ExtEscape. To create watermarks, developers still have to write the necessary values into HKCU Registry. Similarly, developers can control bookmarks by the same escape code. With proper calls, developers can override the bookmark states, e.g., [BookmarkTree] with predefined bookmark node attributes.*

Data Structures

```
#define MAXPATH       256

typedef struct _PDFDocInfo {
       char           title[MAXPATH];              // Title
       char           author[MAXPATH];
       char           creationDate[MAXPATH];
       char           creator[MAXPATH];
       char           producer[MAXPATH];
       char           subject[MAXPATH];
       char           keywords[MAXPATH];
       char           modDate[MAXPATH];
} PDFDocInfo, *PDFDocInfoPtr;

typedef struct _PDFDocInfoW{
       wchar_t        title[MAXPATH];              // Title
       wchar_t        author[MAXPATH];
       wchar_t        creationDate[MAXPATH];
       wchar_t        creator[MAXPATH];
       wchar_t        producer[MAXPATH];
       wchar_t        subject[MAXPATH];
       wchar_t        keywords[MAXPATH];
       wchar_t        modDate[MAXPATH];
} PDFDocInfoW, *PDFDocInfoPtrW;

typedef struct _PDFINPUT             {
       char                  outputfilename[MAXPATH];
       PDFDocInfo            docinfo;
} PDFINPUT, FAR * LPPDFINPUT;

typedef struct _PDFINPUTW            {
       wchar_t               outputfilename[MAXPATH];
       PDFDocInfoW           docinfo;
} PDFINPUTW, FAR * LPPDFINPUTW;
```

Example 1:

```
wchar_t testpasswd[] = L"UserPasswd|poiuy";             // change user passwordd
wchar_t testinfo[] = L"Author|Assign By Program";       // override Author

ExtEscape(pd.hDC, DESC_EPDF_RESET_ALL_KEYVAL, 0, NULL, 0,NULL);
ExtEscape(pd.hDC, PDFFILENAME, sizeof(pdfInput), (LPSTR)&pdfInput, 0, NULL);

ExtEscape(pd.hDC, 5168, wcslen(testpasswd)*2, (LPSTR)testpasswd, 0, NULL);
ExtEscape(pd.hDC, 5168, wcslen(testinfo)*2, (LPSTR)testinfo, 0, NULL);


if (StartDoc (pd.hDC, &di) > 0) {

// where pd is from PrintDlg
```

Example 2:

```
//It utilizes MFC CDC instead. Developers can call this function in OnBeginPrinting.
//Please refer to full source code of the ESCDEMO project.


void EscWDC(CDC *pDC)
{
    PDFINPUTW pdfInput;
    PDFDocInfoW pdfdocinfo;

    memset(&pdfdocinfo,0,sizeof(PDFDocInfo));

    wcscpy(pdfInput.outputfilename, L"c:\\pdfwout.pdf");
    wcscpy(pdfdocinfo.title, L"uni_title");
    wcscpy(pdfdocinfo.author, L"uni_author");
    wcscpy(pdfdocinfo.subject, L"uni_subject");
    wcscpy(pdfdocinfo.keywords, L"uni_keywords");

    pdfInput.docinfo = pdfdocinfo;

    pDC->Escape(DESC_EPDF_RESET_ALL_KEYVAL, 0, NULL, 0,NULL);
    pDC->Escape(DESC_EPDF_PDFFILENAME_UNICODE, sizeof(pdfInput), (LPSTR)&pdfInput,
0,NULL);

}
```

Example 3:

```
//It shows how to enable wait the signal when PDF if fully created by Windows event


void EscWDC(CDC *pDC)
{
    .. .. ..

    wchar_t eventkey[] = L"JobPDFReadyEvent|TestWaitPDFFinishedJob001";
    _pdfeventname = L"TestWaitPDFFinished";


    // Create the event first before starting printing
    _hEvent = CreateEventW(NULL, TRUE, FALSE, _pdfeventname)

    _pdfeventname = L"TestWaitPDFFinished";
    ExtEscape(pDC->m_hDC, 5168, wcslen(eventkey)*2, (LPSTR)eventkey, 0,NULL);

    .. .. ..
}
```

```
void WaitThreadProc(LPVOID lpParam)
{
   DWORD dwWaitResult;

   dwWaitResult = WaitForSingleObject(_hEvent, INFINITE);
   if (dwWaitResult == WAIT_OBJECT_0) {
   .. .. ..

}
```

Tips:

1. The escape codes have to be sent after PrinterDC is ready and before calling StartDoc(). (CREATEDC_POST)

2. The Escape call is multi-job safe in multi-process mode. For issuing multiple jobs from multi-threads in single instance. Developers have to do their own critical section to ensure its safety.

3. Calling DESC_EPDF_RESET_ALL_KEYVAL first to reset all values.

4. To suppress the SaveAs propmpt dialog, developers have to do ExtEscape with PDFFILENAME or DESC_EPDF_PDFFILENAME_UNICODE escape code. DESC_EPDF_PDFFILENAME_UNICODE has the same function with PDFFILENAME except it supports UNICODE fully. If “Deisplay SaveAs Dialog after job spooled” is enabled, developers have to control “DestSaveMode” instead as embedded commands.

5. Escape Code - DESC_EPDF_PERJOB_KEYVAL_UNICODE (5168):

   Currently, all items in [Security], [Doc Info], [Email], and [Destination] can be overridden by this way. Calling ExtEscape with the input buffer pointing to a (wide char *). The wide string is composed as key-value pair with a special character ‘|’.

   For example, setting the key-value pair of user password for encryption will be something like L”UserPasswd|my passwd”. The character ‘|’ is the separator.

6. For key names for the corresponding settings, please refer to the [*Registry Settings*] Section.

Appendix IV: VBA sample for controlling eDocPrinter by registry setting with application level synchronization

Developers may refer this sample for how to set values into registry to control destination PDF filename, suppress SaveAs dialog, and wait for PDF finished event. This is a full working Word VBA Macro sample.

Const:

```
Option Explicit

Public Const EDOC_REG_ROOT = "SOFTWARE\ITEKSOFT\eDocPrinter\PDF\5.0\Profiles\eDocPrinter
PDF Pro\"

Public Const HKEY_CURRENT_USER = &H80000001
Public Const HKEY_LOCAL_MACHINE = &H80000002
Public Const HKEY_USERS = &H80000003

Public Const DELETE = &H10000
Public Const READ_CONTROL = &H20000
Public Const WRITE_DAC = &H40000
Public Const WRITE_OWNER = &H80000
Public Const SYNCHRONIZE = &H100000

Public Const STANDARD_RIGHTS_READ = (READ_CONTROL)
Public Const STANDARD_RIGHTS_WRITE = (READ_CONTROL)
Public Const STANDARD_RIGHTS_EXECUTE = (READ_CONTROL)
Public Const STANDARD_RIGHTS_REQUIRED = &HF0000
Public Const STANDARD_RIGHTS_ALL = &H1F0000

Public Const KEY_QUERY_VALUE = &H1
Public Const KEY_SET_VALUE = &H2
Public Const KEY_CREATE_SUB_KEY = &H4
Public Const KEY_ENUMERATE_SUB_KEYS = &H8
Public Const KEY_NOTIFY = &H10
Public Const KEY_CREATE_LINK = &H20

Public Const REG_SZ = 1             ' Unicode nul terminated string
Public Const INFINITE = &HFFFF      ' Infinite timeout
Public Const STATUS_WAIT_0 = 0
Public Const STATUS_TIMEOUT = 258


Public Const KEY_READ = ((STANDARD_RIGHTS_READ Or KEY_QUERY_VALUE _
                        Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY) And (Not SYNCHRONIZE))
Public Const KEY_WRITE = ((STANDARD_RIGHTS_WRITE Or KEY_SET_VALUE _
                        Or KEY_CREATE_SUB_KEY) And (Not SYNCHRONIZE))
Public Const KEY_ALL_ACCESS = ((STANDARD_RIGHTS_ALL Or KEY_QUERY_VALUE _
                        Or KEY_SET_VALUE Or KEY_CREATE_SUB_KEY _
                        Or KEY_ENUMERATE_SUB_KEYS Or KEY_NOTIFY _
                        Or KEY_CREATE_LINK) And (Not SYNCHRONIZE))
```

Windows API declaration:

```
Declare Function RegSetValue Lib "advapi32.dll" Alias "RegSetValueA" _
    (ByVal hkey As Long, ByVal lpSubKey As String, ByVal dwType As Long, _
     ByVal lpData As String, ByVal cbData As Long) As Long

Declare Function RegSetValueEx Lib "advapi32.dll" Alias "RegSetValueExA" _
    (ByVal hkey As Long, ByVal lpValueName As String, ByVal Reserved As Long, _
     ByVal dwType As Long, lpData As Any, ByVal cbData As Long) As Long

Declare Function RegCreateKeyEx Lib "advapi32.dll" Alias "RegCreateKeyExA" _
    (ByVal hkey As Long, ByVal lpSubKey As String, ByVal Reserved As Long, _
     ByVal lpClass As String, ByVal dwOptions As Long, ByVal samDesired As Long, _
     ByVal lpSecurityAttributes As Long, phkResult As Long, _
     lpdwDisposition As Long) As Long

Declare Function RegCloseKey Lib "advapi32.dll" (ByVal hkey As Long) As Long

Declare Function RegDeleteValue Lib "advapi32.dll" Alias "RegDeleteValueA" _
    (ByVal hkey As Long, ByVal lpValueName As String) As Long

Declare Function RegSetValueExStr Lib "advapi32.dll" Alias "RegSetValueExA" _
    (ByVal hkey As Long, ByVal lpValueName As String, ByVal Reserved As Long, _
     ByVal dwType As Long, ByVal lpData As String, ByVal cbData As Long) As Long

Declare Function CreateEvent Lib "kernel32" Alias "CreateEventA" _
    (lpEventAttributes As Any, ByVal bManualReset As Long, _
    ByVal bInitialState As Long, ByVal lpName As String) As Long

Declare Function ResetEvent Lib "kernel32" (ByVal hEvent As Long) As Long

Declare Function WaitForSingleObject Lib "kernel32" _
    (ByVal hHandle As Long, ByVal dwMilliseconds As Long) As Long

Declare Sub CloseHandle Lib "kernel32" (ByVal hHandle As Long)
```

Helper subroutines:

```
Public Function RegistryOpenKeyForWrite(subkeyname As String, _
        Optional rootname As String = EDOC_REG_ROOT, _
        Optional hRootKey As Long = HKEY_CURRENT_USER) As Long

    Dim hkey As Long
    Dim secaccess As Long

On Error GoTo RegistryOpenKeyForWrite_Error

    secaccess = KEY_WRITE

    If RegCreateKeyEx(hRootKey, rootname & subkeyname, 0, "", 0, _
        secaccess, 0, hkey, 0) = 0 Then

        RegistryOpenKeyForWrite = hkey
    End If

RegistryOpenKeyForWrite_Error:
End Function


Public Function RegistryCloseKey(hkey As Long)
    RegistryCloseKey = RegCloseKey(hkey)

End Function

Public Sub RegistrySetStrValue(hkey As Long, name As String, ByVal val As String)
    If hkey <> 0 Then
        RegSetValueExStr hkey, name, 0, REG_SZ, val, Len(val) + 1
    End If
End Sub
```

Main macro:

```
Sub Demo1()
'
' Demo1 Macro
'

Dim hKeyDest As Long, hEventFinished As Long
Dim pdfokeventName As String
Dim dwWaitResult As Long

    pdfokeventName = "MyPDFJobIsFinished"   ' user defined
    hEventFinished = 0
    hKeyDest = 0
    dwWaitResult = STATUS_TIMEOUT

    ' Create event first
    hEventFinished = CreateEvent(ByVal 0, 0, 0, pdfokeventName)
    If (hEventFinished <> 0) Then
        ResetEvent (hEventFinished)

        hKeyDest = RegistryOpenKeyForWrite("Destination")
        If (hKeyDest <> 0) Then
            Call RegistrySetStrValue(hKeyDest, "JobPDFReadyEvent", pdfokeventName)
            'Un-remark the following if users want to print without saveas dialog
            'Call RegistrySetStrValue(hKeyDest, "DestSaveMode", "1")    ' do not prompt
            'Call RegistrySetStrValue(hKeyDest, "DestExistMode", "1")    ' replace
            'Call RegistrySetStrValue(hKeyDest, "DestDir", "c:\temp\")
            'Call RegistrySetStrValue(hKeyDest, "DestFile", "yourdoc.pdf")

            Call RegistryCloseKey(hKeyDest)
        End If

        ' start printing assume active printer is "eDocPrinter PDF Pro" already
        ' MUST Background:=False as next line will start waiting
        ActiveDocument.PrintOut Background:=False

        ' wait until finished
        ' better waiting in a thread if supported,
        ' or set DestWaitPDFReadyMsg and use RegisterWindowMessage for processing
        Do While (dwWaitResult = STATUS_TIMEOUT)
            dwWaitResult = WaitForSingleObject(hEventFinished, 10)
            DoEvents
            DoEvents
        Loop

        MsgBox "PDF finished"
        CloseHandle (hEventFinished)

        ' clear value se or restore to its original values
        hKeyDest = RegistryOpenKeyForWrite("Destination")
        If (hKeyDest <> 0) Then
            Call RegDeleteValue(hKeyDest, "JobPDFReadyEvent")
            'Delete the values mean reset to default values
            'Or users may read the original values and restore back
            'Call RegDeleteValue(hKeyDest, "DestSaveMode")
            'Call RegDeleteValue(hKeyDest, "DestExistMode")
            'Call RegDeleteValue(hKeyDest, "DestDir")
            'Call RegDeleteValue(hKeyDest, "DestFile")

            RegistryCloseKey (hKeyDest)
        End If

    End If

End Sub
```

Appendix V: VBA sample for controlling eDocPrinter by “eDocPDFAddin.Control” COM methods

Since Ver 6.24, a COM with class name “eDocPDFAddin.Control” is provided as part of ADV pack for developers to control eDocPrinter with registry settings with handy methods without calling Windows API directly. Below is a Word VBA Macro sample working the same as previous one but using the COM methods. (The COM DLL is epdfcom.dll)

Methods:

```
Public Sub SelectProfile(profilename As String)
Public Sub WriteKeyStr(category As String, keyname As String, value As String, Optional
profilename As String = "-1")

Public Sub PreparePDFEvent(Optional profilename As String = "-1", Optional eventname As
String = "-1")
Public Sub WaitPDFFinished()
```

Sample Program:

```
Sub Demo2()
    Dim epdf As Object

    Set epdf = CreateObject("eDocPDFAddin.Control")

    If Not (epdf Is Nothing) Then

        Call epdf.SelectProfile("eDocPrinter PDF Pro")    ' select the profile to modify

        'Call epdf.WriteKeyStr("Destination", "DestSaveMode", "1")    ' do not prompt
        'Call epdf.WriteKeyStr("Destination", "DestExistMode", "1")    ' replace
        'Call epdf.WriteKeyStr("Destination", "DestDir", "c:\temp\")
        'Call epdf.WriteKeyStr("Destination", "DestFile", "yourdoc.pdf")

        epdf.PreparePDFEvent  ' do all necessary event preparation
        ' MUST Background:=False as next line will start waiting
        ActiveDocument.PrintOut Background:=False

        epdf.WaitPDFFinished  ' will wait until job finished


        'Call epdf.WriteKeyStr("Destination", "DestSaveMode", "0")    ' restore values
        'Call epdf.WriteKeyStr("Destination", "DestExistMode", "0")
        'Call epdf.WriteKeyStr("Destination", "DestDir", "")
        'Call epdf.WriteKeyStr("Destination", "DestFile", "")

    End If

End Sub
```