



SMARTPRINTER

Reference Manual

Version 1.9.0

Smart Printing Solutions LTD

November 2022

All rights reserved. Neither this documentation nor any part of it may be reproduced, stored in a retrieval system, translated into another language, or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of Smart Printing Solutions (SPS) LTD.

While every precaution has been taken in the preparation of this manual, SPS assumes no responsibility for errors or omissions; neither any liability is assumed for damages resulting from the use of the information contained herein. The information contained in this document is subject to change without notice. SPS reserves the rights to make any such changes without obligation to notify any person of such revision or changes. SPS makes no commitment to keep the information contained herein up to date.

Copyright © 2001-2022 Smart Printing Solutions (SPS) LTD.

Address: 12 Hanapach Street, P.O.Box 1109, Karmiel

Zip 2165318, ISRAEL

Tel: +972-4-9909357

Fax: +972-4-9990068

Email: sales@sps.co.il

Web: www.sps.co.il

SmartPrinter Introduction

SmartPrinter is a smart print spooler. It was created in order to provide several types of printing solutions:

- * Language support (e.g. Hebrew, Greek, Turkish, etc...)
- * Printer emulations support (e.g. Epson, DEC, IBM Proprinter, etc...)
- * Preprinted forms and templates
- * Special customized solutions

The SmartPrinter software contains several components:

- * [SmartPrinter PrintProcessor](#)
- * [SmartPrinter PrintController](#)
- * [SmartPrinter Event Log Viewer](#)
- * [SmartPrinter LPD Server](#)

Please see [SmartPrinter process procedure](#) in order to understand more about the way SmartPrinter process print jobs.

Please see the [SmartPrinter system requirements](#), to find out weather the hardware you possess is suitable for use with SmartPrinter.

SmartPrinter System Requirements

SmartPrinter software can receive a print job from any operating system. However, SmartPrinter software is installed in a computer running a Windows operating system from one of the following versions: XP / 2000 / 2003 / NT 4.0 / 98. The recommended operating systems are Windows XP or Windows 2003 Server.

The hardware on which the SmartPrinter software is installed must match the load of print jobs the software needs to handle. Generally a Pentium 4 processor with 128MB of RAM is sufficient for most installations.



during the standard operation of the software, mainly when working with large print jobs, there is a lot of use in cache memory, therefore hardware based on Celeron processor is not recommended for working under heavy load.

SmartPrinter software is destined only for printers supporting PCL5 printing language

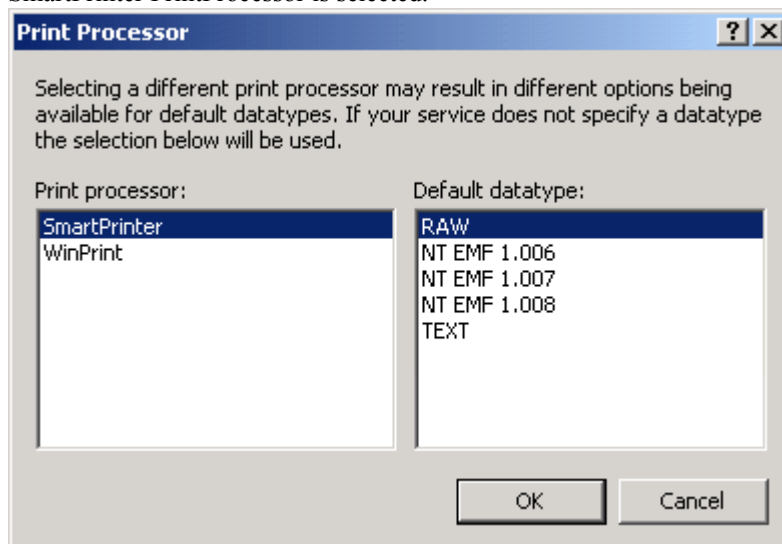
SmartPrinter PrintProcessor

The PrintProcessor is the heart of the SmartPrinter system. Any printer using the SmartPrinter PrintProcessor, will become a special printer which will follow the [SmartPrinter print process procedure](#)

In order to establish whether a certain printer is using the SmartPrinter PrintProcessor, please follow the procedure below:

For Windows NT/2000/XP/2003:

- 1 Click <Start> | <Settings> | <Printers>.
- 2 The Printers folder will open. Select the desired printer.
- 3 Right Click on it, and select Properties.
- 4 Select the Advanced tab.
- 5 Click <Print Processor...> the following dialog box will be opened, make sure that the SmartPrinter PrintProcessor is selected.



For Windows 95/98/ME:

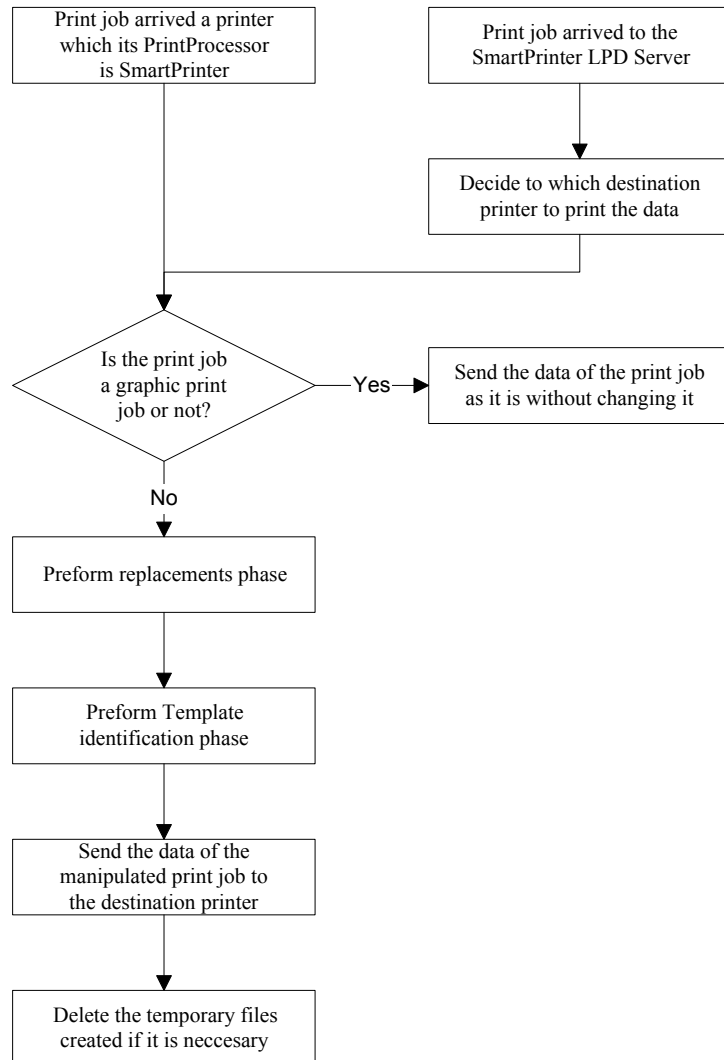
There is no direct way to find out which PrintProcessor a certain printer uses in windows 98/95/ME. The only way to know it is through the registry itself. It is not recommended to handle the registry file manually.



The standard (default) windows PrintProcessor is WinPrint. In order to restore a printer that was converted into a SmartPrinter printer back to its original state, you should simply change back the PrintProcessor to be WinPrint.

SmartPrinter Process Procedure

This section describes the process a print jobs goes through when it is being printed using a SmartPrinter printer.



Steps description:

* **Receive job**

The print job is received and ready to be processed, there are 3 [ways to receive](#) the print job into SmartPrinter

* **Job type identification**

The SmartPrinter decides weather the print job is a graphic print job or not. If the print job is a graphic print job, the SmartPrinter may pass it through without touching it, so that the printer will produce the exact same printout as a regular printer would have produced. If the job is not a graphic print job, it continues to the next step.

* **Replace phase**

In this phase, the SmartPrinter will use the rules specified in the active [replace file](#) to manipulate the print job

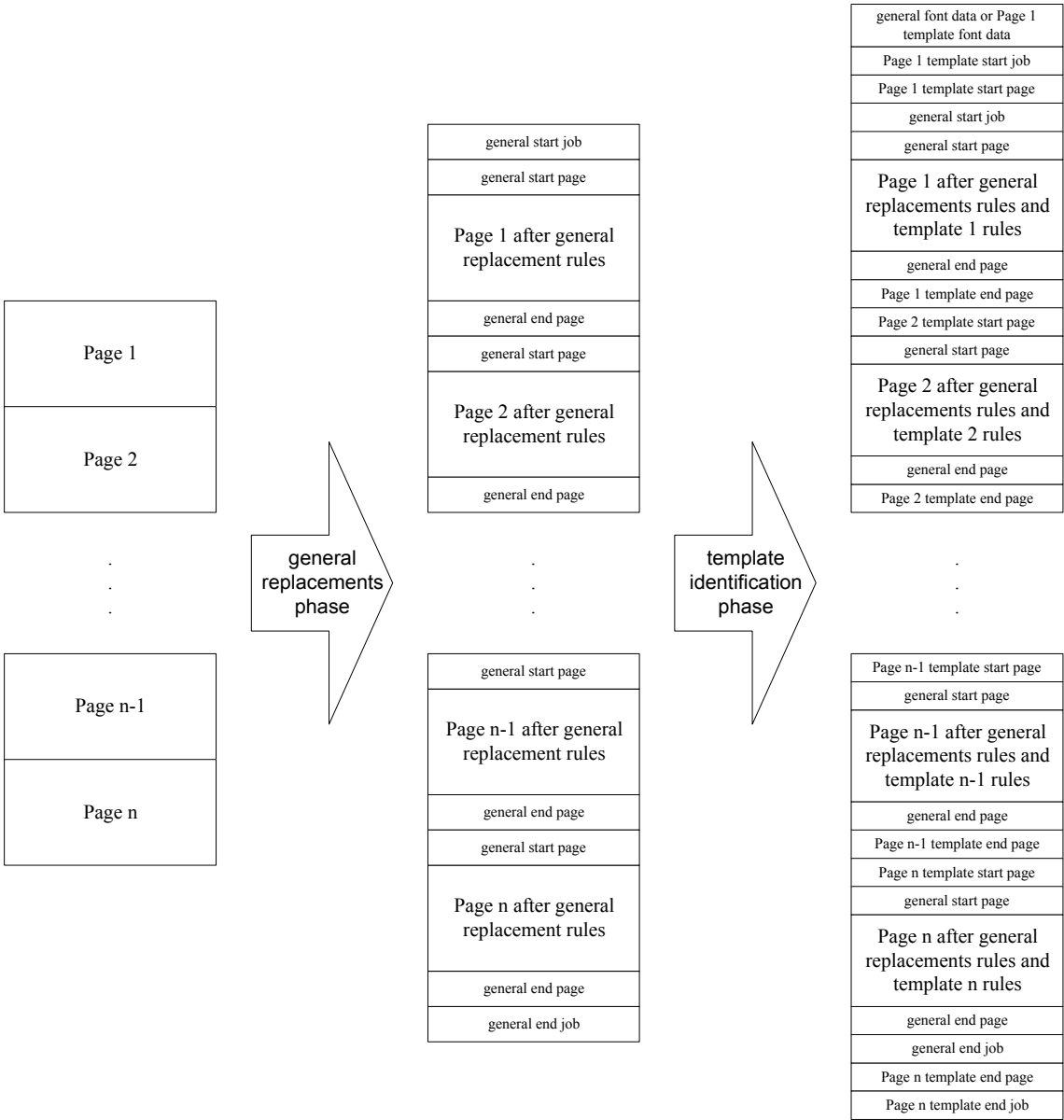
* **Template Identification phase**

In this phase the SmartPrinter will search for a [template file](#) matches a Key exists in the

print job. If such a key was found, the appropriate template file will be selected and the SmartPrinter will use the rules and [script functions](#) specified in the template file to manipulate the print job.

- * **Print phase**
In this phase the manipulated print job will be sent to the printer instead of the original print job.
- * **Delete phase**
In this phase all of the temporary files (including the original print job, the manipulated print job, etc...) will be deleted.

The following scheme describes the effects the general replacements phase and the template identification phase has on the print job:



SmartPrinter Receive ways

There are 3 ways to receive a print job into the SmartPrinter system:

1. Windows Printer

When a windows printer's PrintProcessor is changed from WinPrint to SmartPrinter, any job that will be printed to that printer will go through the [SmartPrinter process procedure](#). You can share the above printer, and print to it from any computer and any application. You can also capture the shared printer on the remote computer in order to print DOS print jobs via the SmartPrinter.



If you print via windows printer from DOS like application and from windows graphic application, make sure to check the “**Print jobs as RAW if it starts with the UEL command**” checkbox under the [Options dialog box](#) in the PrintController. This will instruct SmartPrinter to handle only the non-graphic print jobs.

2. LPD Server

The LPD server is designed to receive print jobs from hosts that use the LPR-LPD printing protocol. Please refer to [SmartPrinter LPD Server](#) for more information.



If you have installed the SmartPrinter on a computer running Windows NT / 2000 / XP / 2003 operating system, it is also possible to use the Microsoft TCP/IP Print Server Service in order to receive print jobs from the host, please refer to [Using Microsoft TCP/IP Print Server Service](#) for more information.

3. Hot Directory

Hot Directory is a method in which a certain folder is being scanned every time period for new files, if there are new files, they will be processed by the SmartPrinter, and then deleted. Please refer to [hot directory dialog box](#) in the PrintController for more information



Hot Directory is not a recommended way to receive print jobs, since there may be a situation in which a file is written on top of an older file before SmartPrinter had a chance to read the older file. If this is the only possible way for receiving print jobs from your application into SmartPrinter, you must have a certain control mechanism over the names of the files in the hot directory, in order to prevent this problem.

SmartPrinter LPD Server

The SmartPrinter LPD server is used if you wish to receive print jobs from sources printing through an LPR-LPD connection, sources such as Unix, Mainframe, VMS, etc... These systems send their print jobs using an LPR command, and expect the printer to have an LPD server in order to receive the print job.

In order to configure the SmartPrinter to receive print jobs through its LPD server, you should follow these steps:

- 1 Start the SMPLPD.EXE application. The following icon should appear on your windows systray



- 2 In your host server add a printer or remote queue, which its destination IP address is the IP address of the windows running the SmartPrinter LPD server, and the remote queue name is the name of the windows printer you wish the print job will be printed on.

LPD Server configuration:

The settings and configurations for the LPD server are specified in the [internal LPD dialog box](#) in the PrintController.



The way to create a printer or a remote queue, is different from one host to another, therefore it is impossible to specify here how to add it at any environment. The system administrator that is responsible to the specific host should have that knowledge.



If your SmartPrinter software is installed on a computer running Windows NT/2000/XP/2003 operating systems, you may wish to work with the [Microsoft LPD Server](#), or the TCP/IP Print Server Service supplied by Microsoft.



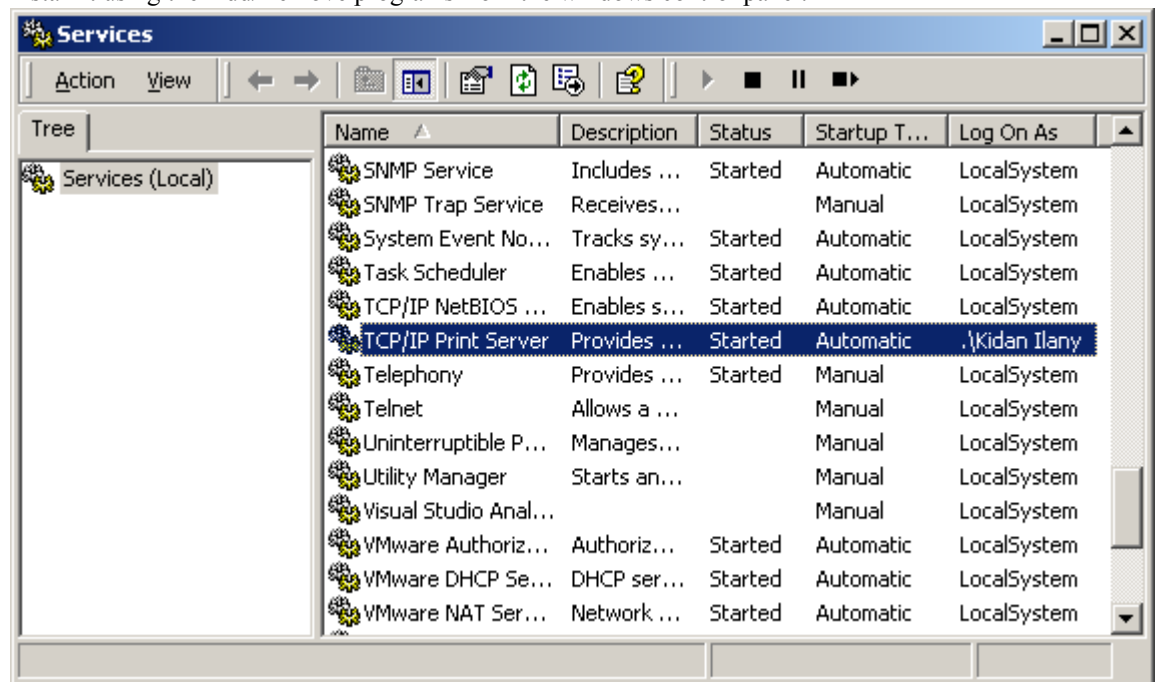
There is a possibility to make a connection between the remote queue name on the source server and the printer to which the print job will be sent using the [Printer Swapping dialog box](#) in the PrintController. This is useful when you must stick with a certain name convention on the source server and a different name convention on the windows computer on which the SmartPrinter is installed.

Using Microsoft TCP/IP Print Server Service

In addition to working with the [SmartPrinter LPD server](#), SmartPrinter is also compatible with the Microsoft LPD Server, or the TCP/IP Print Server Service supplied by Microsoft. This service exists only on Windows NT/2000/XP/2003 operating systems and not on Windows 95/98/ME.

In order to configure the SmartPrinter to receive print jobs through the Microsoft LPD server, you should follow these steps:

- 1 Start the TCP/IP Print Server Service from the services of the computer on which the SmartPrinter is installed. If the service doesn't exist in your list of services, you need to install it using the Add/Remove programs from the windows control panel.



- 2 In your host server add a printer or remote queue, which its destination IP address is the IP address of the windows running the SmartPrinter LPD server, and the remote queue name is identical to the name of the printer on which the SmartPrinter PrintProcessor is installed.



Unlike the SmartPrinter LPD Server, in which all of the queue names sent to the LPD server are captured, and there is a possibility to make a connection between the remote queue name and the printer to which the print job will be sent using the [Printer Swapping dialog box](#) in the PrintController. When you use the Microsoft LPD server, the remote queue name on the host must be exactly the same as the printer name.

PrintController Introduction

The PrintController is an application that controls all of the SmartPrinter parameters. Here you can manage print jobs, set which parameters and files the SmartPrinter will use during the print process, control the SmartPrinter LPD server, select which fonts to use, select which are the licensed printers, etc...

The main window of the PrintController is the [Job Manager](#).

PrintController Operations

The following menus are available in the PrintController interface:

- * **[File]**
 - * **[Reprint job]**
Activate the [Reprint job dialog box](#), which enables you to reprint any of the files related to the job to any licensed printer.
 - * **View job**
This option will only be available if you have defined an external PCL viewer in the [Options dialog box](#), when selected, it will activate the external PCL viewer with the current job file, this will allow you to view your print job.
 - * **[Delete job files]**
Delete all marked print jobs and all the temporary files related to it.
 - * **[Rename job]**
Changes the name of the print job.
 - * **View job information...**
Opens the [View job Information dialog box](#) which shows you all the relevant information regarding the print job.
 - * **[Print External File]**
Activate the [Print external file dialog box](#), which enables you to print any file any licensed printer.
 - * **[Empty Spool]**
Delete all saved print jobs from the job manager.
 - * **[Save print jobs]**
When this menu is checked, some of the temporary files created during the job manipulation process are saved after the manipulated job was printed. When this menu is unchecked, these temporary files are deleted.
In order to select which temporary files will be deleted and which will not be deleted, click the Tools | Options menu.
 - * **[Exit]**
This closes the PrintController

* **[View]**

* **[Toolbar]**

When this menu is checked, the toolbar will be displayed. The toolbar allows you to quickly access different menu selections.

* **[Status bar]**

When this menu is checked, the status bar will be displayed. The status bar shows you different information regarding the jobs in the PrintController.

* **[Hide in systray]**

When you click this menu, the PrintController will be hidden in the systray.



You can restore it from there by right clicking the icon and select Restore from the popup menu.

* **[Event log...]**

This menu will start the [event log viewer](#).

* **[Sort By]**

Using this menu you can sort the jobs in the manager, the sort can be done according to any field in the manager, and can be either ascending or descending. You can also sort the jobs by clicking the header of the desired column to sort by.

* **[Select All]**

This menu causes all the jobs in the job manager to be selected.

* **[Refresh]**

This menu causes the job manager to be refreshed, and to read the saved print jobs from the spool directory.

* **[Manage]**

* **[Fonts...]**

This menu will start the [Font management dialog box](#).

* **[Translation tables...]**

This menu will start the [Translation tables management dialog box](#).

* **[GDI]**

This menu will start the [GDI Print Parameters dialog box](#).

* **[GDI Parameters...]**

This menu will start the [GDI Print Parameters dialog box](#).

* **[Internal PDF writer settings...]**

This menu will start the [Internal PDF writer settings dialog box](#).

*** [Email settings...]**

This menu will start the [Email settings dialog box](#).

*** [Templates...]**

This menu will start the [Template management dialog box](#), which enables you to select a template to be edited, and edit it.

*** [Replace files...]**

This menu will start the [Replace files management dialog box](#), which enables you to select a replace file to be edited, and edit it.

*** [Emulation conversions...]**

This menu will start the Emulation conversions parameters dialog box, which enables you to change the parameters related to emulation conversion.

The following print emulations are supported:

[Epson ESC/P](#), [Digital Dec](#), [Canon CaPSL](#), [Kyocera Prescribe](#), [Brother barcodes](#)

*** [Tools]**

*** [Options...]**

This menu will start the [Options dialog box](#), which enables you to control different parameters of the SmartPrinter.

*** [Printer swapping]**

This menu will start the [Printer Swapping dialog box](#).

*** [Last job number...]**

This menu will start the [Last job number dialog box](#).

*** [SmartPrinter file converter...]**

This menu will start the [SmartPrinter file converter dialog box](#).

*** [Internal LPD server]**

This menu will start the [Internal LPD Server dialog box](#).

*** [Hot Directory]**

This menu will start the [Hot Directory dialog box](#).

*** [SNMP Options]**

This menu will start the [SNMP Options dialog box](#).

*** [New license code]**

When you click this menu, the [license dialog box](#) opens, and you can change your SmartPrinter license code.

*** [Licensed Printers]**

When you click this menu, the [Licensed Printers dialog box](#) opens, and you can

select the licensed printers. In addition you can create and edit printer groups.

- * **[Users Lists]**

This menu will start the [Users Lists dialog box](#).

- * **[PrintController password]**

When you click this menu, the [PrintController Password dialog box](#) opens, and you can change your PrintController operation password.

- * **[Help]**

- * **[Introduction]**

This menu shows this help topic (PrintController Introduction).

- * **[Contents]**

This menu shows the content of the help.

- * **[Search help on]**

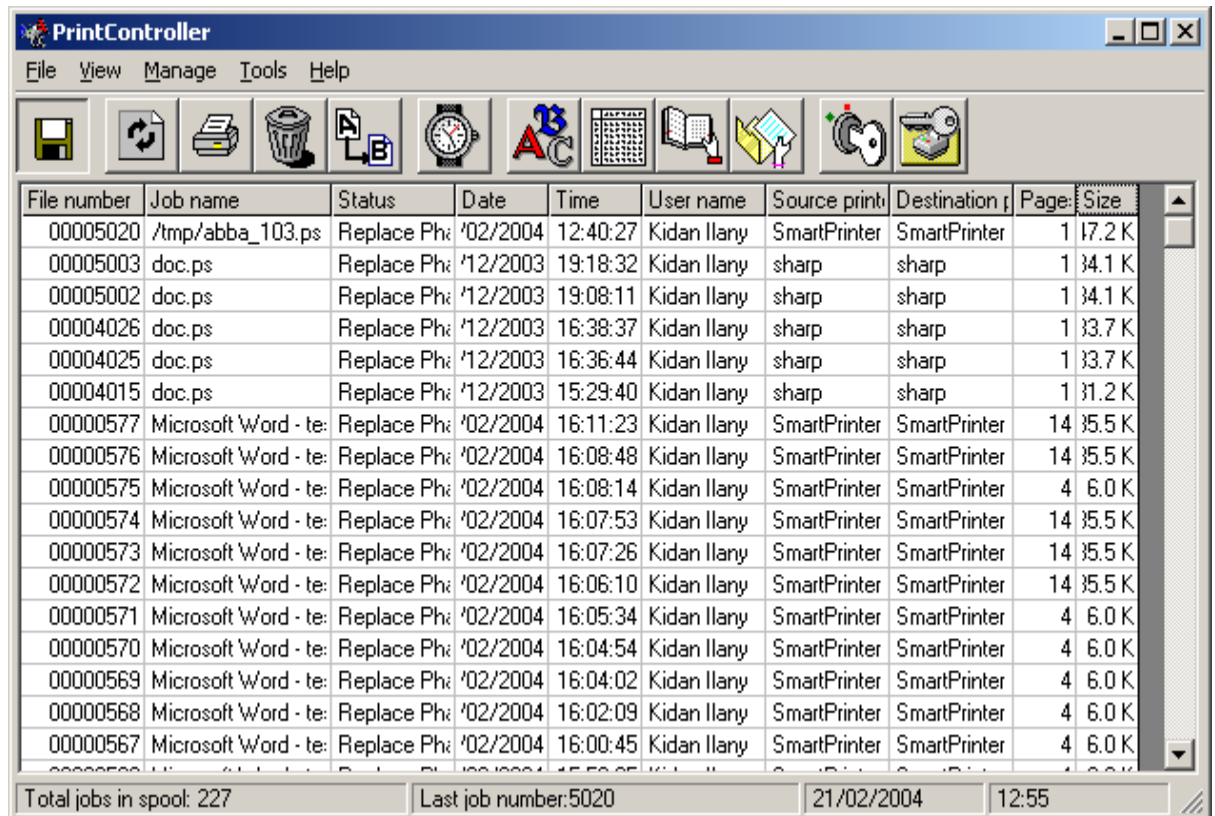
This menu starts the search for topics in the help.

- * **[About]**

This menu shows the about dialog box. In this dialog box you can view the SmartPrinter version and registration information.

PrintController Job Manager

The PrintController is the place where you can manage your old print jobs (print jobs which were printed and their temporary files were saved). You can see all of the old jobs and the properties of these print jobs.



File number	Job name	Status	Date	Time	User name	Source printer	Destination printer	Page	Size
00005020	/tmp/abba_103.ps	Replace Phi	'02/2004	12:40:27	Kidan Ilany	SmartPrinter	SmartPrinter	1	17.2 K
00005003	doc.ps	Replace Phi	'12/2003	19:18:32	Kidan Ilany	sharp	sharp	1	14.1 K
00005002	doc.ps	Replace Phi	'12/2003	19:08:11	Kidan Ilany	sharp	sharp	1	14.1 K
00004026	doc.ps	Replace Phi	'12/2003	16:38:37	Kidan Ilany	sharp	sharp	1	13.7 K
00004025	doc.ps	Replace Phi	'12/2003	16:36:44	Kidan Ilany	sharp	sharp	1	13.7 K
00004015	doc.ps	Replace Phi	'12/2003	15:29:40	Kidan Ilany	sharp	sharp	1	11.2 K
00000577	Microsoft Word - te	Replace Phi	'02/2004	16:11:23	Kidan Ilany	SmartPrinter	SmartPrinter	14	15.5 K
00000576	Microsoft Word - te	Replace Phi	'02/2004	16:08:48	Kidan Ilany	SmartPrinter	SmartPrinter	14	15.5 K
00000575	Microsoft Word - te	Replace Phi	'02/2004	16:08:14	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K
00000574	Microsoft Word - te	Replace Phi	'02/2004	16:07:53	Kidan Ilany	SmartPrinter	SmartPrinter	14	15.5 K
00000573	Microsoft Word - te	Replace Phi	'02/2004	16:07:26	Kidan Ilany	SmartPrinter	SmartPrinter	14	15.5 K
00000572	Microsoft Word - te	Replace Phi	'02/2004	16:06:10	Kidan Ilany	SmartPrinter	SmartPrinter	14	15.5 K
00000571	Microsoft Word - te	Replace Phi	'02/2004	16:05:34	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K
00000570	Microsoft Word - te	Replace Phi	'02/2004	16:04:54	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K
00000569	Microsoft Word - te	Replace Phi	'02/2004	16:04:02	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K
00000568	Microsoft Word - te	Replace Phi	'02/2004	16:02:09	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K
00000567	Microsoft Word - te	Replace Phi	'02/2004	16:00:45	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K
00000566	Microsoft Word - te	Replace Phi	'02/2004	15:58:05	Kidan Ilany	SmartPrinter	SmartPrinter	4	6.0 K

Total jobs in spool: 227 Last job number: 5020 21/02/2004 12:55

Print job properties

The following properties are available for each job in the job manager:

- * **File Number**

The number of the file containing the job, the meaning is for the actual temporary file in the SmartPrinter spool folder.

- * **Job Name**

The name of the print job as it arrived into the SmartPrinter printer or to the LPD server.

- * **Status**

The status of the print job

- * **Date**

The date in which the print job arrived into the job manager.

- * **Time**

The time in which the print job arrived into the job manager.

* **User Name**

The name of the user who sent this print job.

* **Source**

The name of the printer to which the print job was printed or the name of the queue in the LPD server to which the print job was sent.

* **Destination**

The name of the printer to which the print job will be eventually printed.

* **Pages**

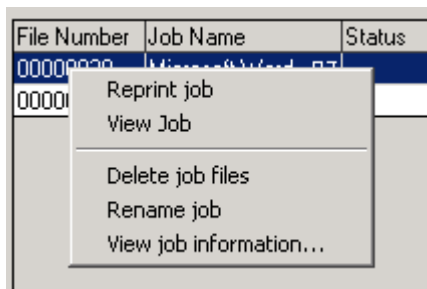
The number of pages in the print job

* **Size**

The size of the print job in bytes.

Job Manager Operations

To make an operation on a specific job, you should select the job, then right click it and select an action to perform. You can also perform the same actions from the file menu or from the toolbar. The following actions are available:



* **Reprint job**

Activate the [Reprint job dialog box](#), which enables you to reprint any of the files related to the job to any licensed printer.

* **View job**

This option will only be available if you have defined an external PCL viewer in the [Options dialog box](#), when selected, it will activate the external PCL viewer with the current job file, this will allow you to view your print job.

* **Delete job files**

Delete the print job and all the temporary files related to it.

* **Rename job**

Changes the name of the print job.

* **View job information...**

Opens the [View job Information dialog box](#) which shows you all the relevant information regarding the print job.



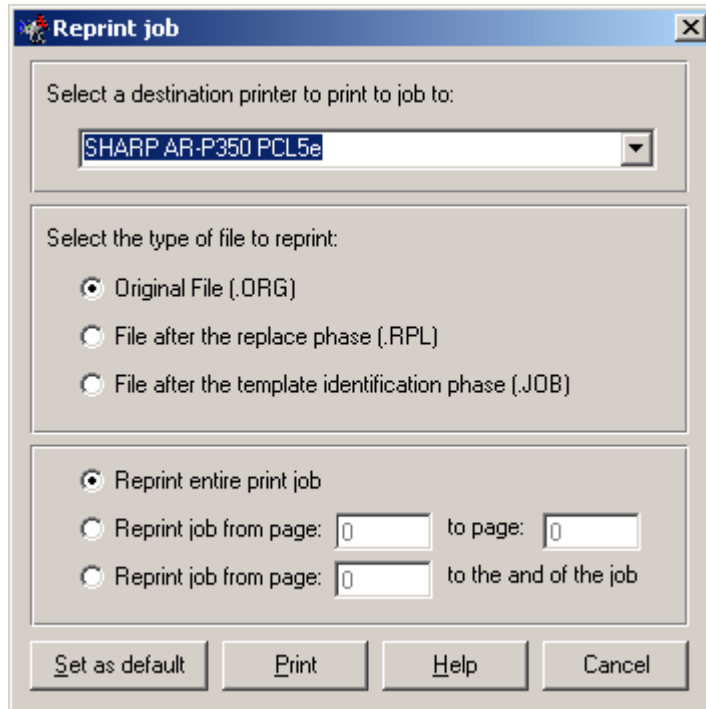
If the print job arrived from a windows printer that its PrintProcessor is SmartPrinter then the source and destination printers are the same and they are both the above printer.

If the print job arrived from the LPD server then the source is the queue name as it appears on the host server and the destination is the windows printer to which the manipulated print job will be printed. The connection between the source and the destination is made in the [Printer Swapping dialog box](#).

PrintController Reprint job Dialog Box

The Reprint job dialog box is the place where you can perform a reprint of a job or a part of a job.

You may print any of the temporary files related to the print job to any licensed printer.



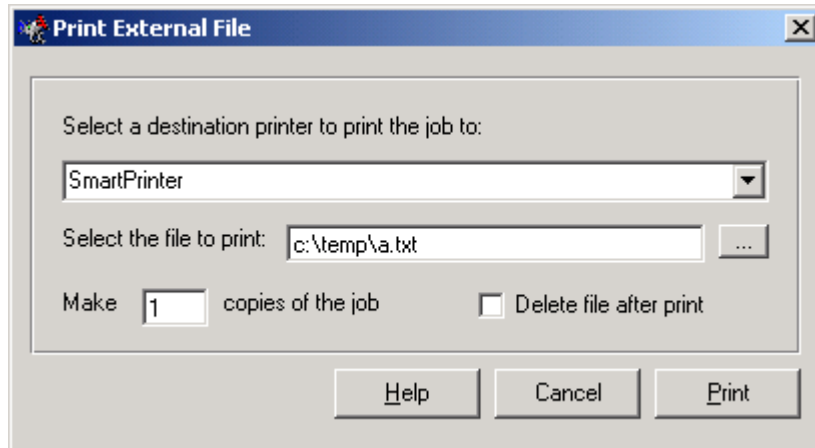
In order to reprint a job:

1. Select the destination printer from the “printers combo box”.
2. Select the type of file to reprint from the options enabled in the radio buttons.
3. If you wish to reprint the entire job, select the “Reprint entire print job”, if you wish to print only a portion of the job, select the second option, and type in the pages range to be printed.
4. Click the <Print> button.

You may click the <Set as default> button in order to save the current settings, then in the next time you will reprint a job, the settings you have saved will be the default one.

PrintController Print External File Dialog Box

The print external file dialog box is the place where you can print any file to any licensed printer on your computer. This is useful when you have files you have received offline and you want to print it to a SmartPrinter printer.



In order to print a file:

1. Select the destination printer from the “printers combo box”.
2. Select the file you want to print.
3. Select how many time you want the file to be printed (default = 1 copy).
4. If you wish that the file will be deleted after the print is ended, check the appropriate checkbox.
5. Click the <Print> button.

PrintController View Job Information Dialog Box

The view job information dialog box is where you can view all information relevant to the selected print job.

Job name:	disenhouse	Job size:	2.2 K
Job file number:	16178	Total pages:	1
Date spooled:	15/05/2004	User name:	Kidani Ilany
Time spooled:	13:24:38	Source printer:	smartp
Job status:	ID Phase Ended	Destination printer:	smartp

File used during replace phase: c:\smartprinter\replace\epson.rpl

Templates matched to each page of the job:

Page 1: c:\smartprinter\templates\eliyahu.LCS

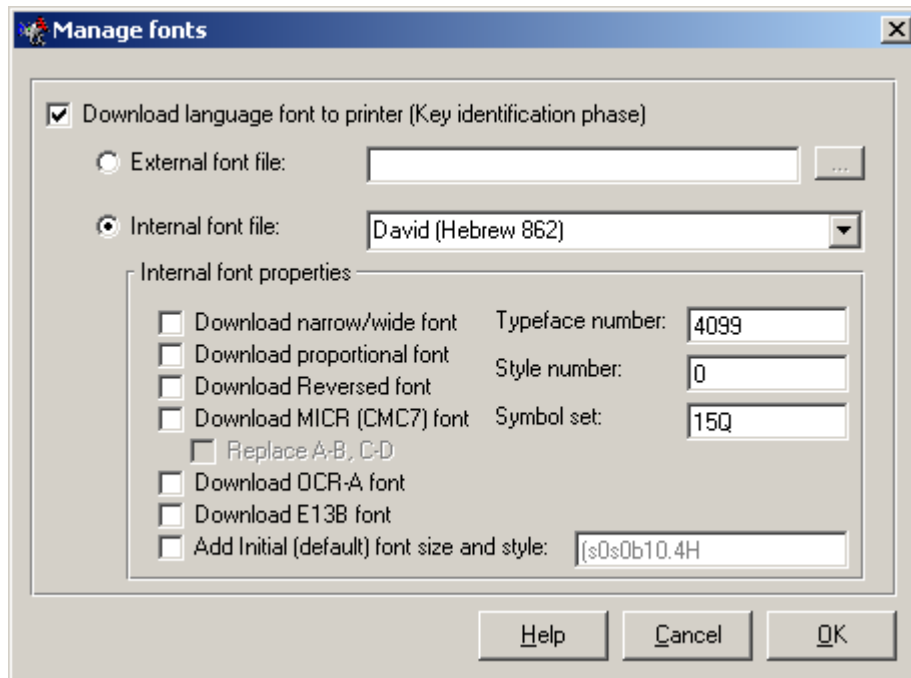
Help OK

The information available here includes data such as: job name, sender, time and date, etc...

You can also see which replace file was actually used in the Replace Phase, and which template was used for each page in the Template Identification Phase.

PrintController Font Management Dialog Box

The Font management dialog box contains all the parameters that controls the fonts:



* **Download language font to printer**

When this checkbox is checked, the SmartPrinter will download the selected font file to the printer. The font data will come before the start job parameter. The font addition to the print job occurs in the [template identification phase](#), it means that in the template identification phase the print job doesn't contain the font yet.

* **Internal Font File**

Internal font file is a build-in font, which can be used easily by selecting the preferred font from the combo box. When you elect an internal font, the internal font properties opens, and you can set these properties:

* **Download Narrow/Wide font**

This property allows you to choose whether you like to download narrow and wide fonts in addition to the standard font.



When an internal font which is "Wide-Narrow" is selected, you can activate these functionality by using the style of the font, when the Narrow is style 8 (Activated by inserting the command "<ESC>(s8S") and the Wide is style 24 (Activated by inserting the command "<ESC>(s24S").

* **Download Proportional font**

This property allows you to choose whether you like to download proportional fonts in addition to the standard

fixed font.

* **Download Reversed font**

This property allows you to choose whether you like to download reversed (white over black background) fonts in addition to the standard font.



When an internal font which is “Reversed” is selected, you can activate these functionality by using the style of the font, when the Reversed is style 512 (Activated by inserting the command “<ESC>(s512S”).

* **Download MICR (CMC7) font**

This property allows you to choose whether you like to download MICR (CMC7) font, this font is used for payable documents, such as checks and vouchers. You can load this for with or without the A-B, C-D character switching



When the internal MICR font is selected, you can activate it by inserting the command <ESC>(8M<ESC>(s1p12v0s4b4457T”.

* **Download OCR-A font**

This property allows you to choose whether you like to download OCR-A font, this font is used for payable documents, such as checks and vouchers.



When the internal OCR-A font is selected, you can activate it by inserting the command <ESC>(0O<ESC>(s0p10h0s0b4200T”.

* **Download MICR (E13B) font**

This property allows you to choose whether you like to download MICR (E13B) font, this font is used for payable documents, such as checks and vouchers. You can load this for with or without the A-B, C-D character switching



When the internal MICR font is selected, you can activate it by inserting the command <ESC>(8M<ESC>(s1p12v0s4b4457T”.

* **Add initial (Default) font size and style**

This property allows you to choose the initial font size and style that will be chosen immediately after the font is loaded to the printer. Any text exists in the input data file before a new font size command is issued will be printed in this size and style.

* **Typeface Number**

This property allows you to choose what typeface number would you like the font to have.

* **Style number**

This property allows you to choose what style number would you like the font to have.

* **Symbol set**

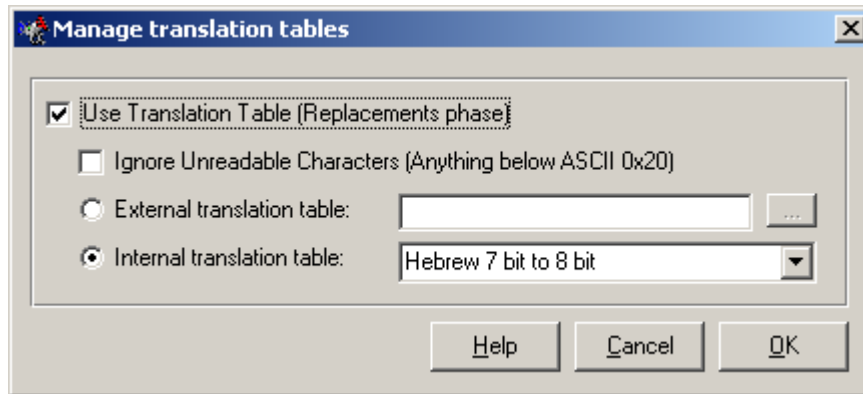
This property allows you to choose what character set would you like the font to have.

* **External Font File**

External font file can be any font that is supported by your printer.

PrintController Translation Table Management Dialog Box

The translation table management dialog box contains all the parameters that controls the language translation tables:



*** Use Translation Table**

When this checkbox is checked, the SmartPrinter will translate the print job according to the selected translation table. The translation will include all characters that are not inside a PCL command (PCL commands will not be translated).

The character translation occurs in the replace phase, it means that in the template identification phase the print job is already translated.

*** Ignore Unreadable Characters (Anything below ASCII 0x20)**

When this checkbox is checked, and a translation table of any type (internal or external) is in use, the SmartPrinter will ignore any character that is below the Hex value of 0x20 (Space character) and will translate it into space.

*** Internal Translation Table**

Internal translation table is a build-in translation-table, which is exactly the same as an external translation table.

*** External Translation Table**

External translation table is a translation table that is written in a TBL file. A TBL file is a binary file contains instructions for the translation of the characters, the structure of a TBL file is as follows:

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
10	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
20	20	21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
30	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F
40	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
50	50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
60	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F
70	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E	7F
80	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
F0	90	91	92	93	94	95	96	97	98	99	9A	FB	FC	FD	FE	FF

In the above example, the character that its ASCII is "E0" is substituted by the character that its ASCII is "80". That is because in the bit number "E0" of the file appears the character that its ASCII is "80". In addition, the character "E1" becomes "81", and so on until the character "FA" that becomes "9A".

PrintController GDI Print Parameters Dialog Box

GDI Parameters

General Settings

- ☐ Process PCL job macros
- ☐ Start with new page
- Gdi Default Extension:
- Default resolution:

Image Printers

PDF Printer:

PDF Writer Type: ☐ External ☒ Internal

TIFF Printer:

Font Parameters

Code Page:

Regular Font Weight:

Bold Font Weight:

Tall Font Height Percent:

Wide Font Width Percent:

Margin Adjustment

	Top	Left
GDI Printer:	<input type="text" value="0"/>	<input type="text" value="0"/>
FAX:	<input type="text" value="0"/>	<input type="text" value="0"/>
PDF:	<input type="text" value="0"/>	<input type="text" value="0"/>
TIFF:	<input type="text" value="0"/>	<input type="text" value="0"/>

EMF Macro Definitions

Fax & Email Parameters

Default Fax Name:

Default Fax Number:

The GDI Print Parameters dialog box is used to control the conversion of PCL print jobs, after the Replace and Template phases, into Windows GDI (Graphic Driver Interface). This is done in order to use the original printer driver in the final job building process. The result is that the job can be printed on any printer, even printers not supporting PCL5.

The available options are:

* **Process PCL job macros**

When this checkbox is checked, the SmartPrinter will translate PCL macros in the PCL job. When this checkbox is not checked, the SmartPrinter will only use EMF macros as a replacement to the PCL Macros.

* **Start with new page**

When this checkbox is checked, the SmartPrinter will issue a new page command in the beginning of the print job.

* **GDI default extension**

When this parameter is used, the SmartPrinter will forward the print job to be printed on a different printer than the one the print job arrived in. For example, if the parameter is set to be "w", and a print job arrived at the SmartPrinter, whose name is "SmartP", then the SmartPrinter will actually print the job to a printer called "SmartPw".

* **Default Resolution**

This parameter sets the default resolution for the PCL device that the original file was designed to.

* **Image Printers**

This parameter sets the Printer driver through which the PDF and TIFF printing will be made.

If you are using the internal PDF writer, you need to select a Postscript printer here, that its port is a local dummy port. You can also choose more options under the [Internal PDF writer settings dialog box](#).

* **Fax & Email Parameters**

These parameters set the default fax name and number to send the fax to in case the #SetFaxNumber() and #SetFaxName script functions were not properly used in the template. In here you can also set the email parameters through the [Email settings dialog box](#).

* **Top and Left Margin adjustment**

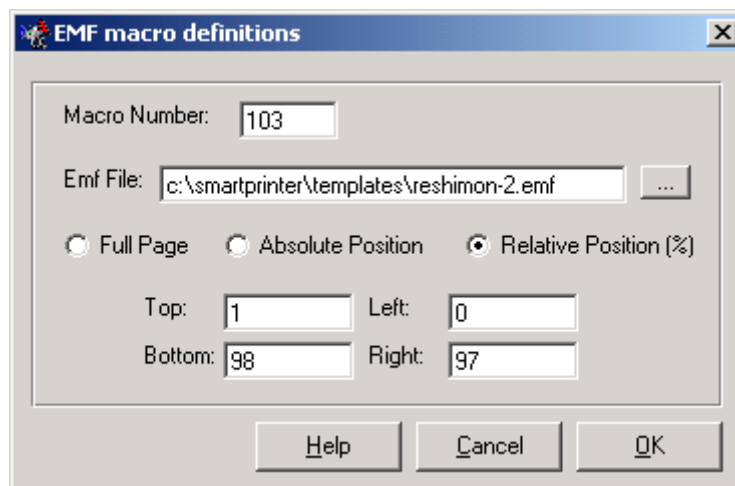
This parameter sets the top and left margins of the GDI translation. Each type of device (GDI Printer, Fax, PDF and TIFF) can have a different margin setup.

* **Font Parameters**

These parameters set the font properties. The Code Page Combo box decides which font code page will be in use, the Bold and Regular font weight sets the weight of the font, and the Height and Width percent decides by how much percent the font size will be multiplied when Tall and Wide fonts are in use.

* **EMF macro definitions**

Here you can set the EMF files that will be used in the translation instead of the PCL macro calls in the PCL print job. Each EMF macro has definitions that can be selected by clicking the <EDIT>



* **Macro Number**

This parameter sets the number of the macro that will be replaced with the EMF file.

* **EMF File**

This parameter sets the EMF file that the macro number will be replaced with.

* **Position and size of the macro**

This parameter sets where will the EMF file be positioned on the page, there are 3 options:

1. **Full Page**

The EMF will be printed as a background picture of the entire page

2. **Absolute Position**

The EMF will be printed in the position written in the Top, Left, Bottom, Right fields, where the units of these fields are in device units (e.g. if the GDI printer is a 600 DPI printer the units will be 600 DPI units).

3. **Relative Position**

The EMF will be printed in the position written in the Top, Left, Bottom, Right fields, where the units of these fields are in percentage of the page (e.g. a relative position of Top=0, Left=0, Bottom=100, Right=100 equals exactly to Full Page).

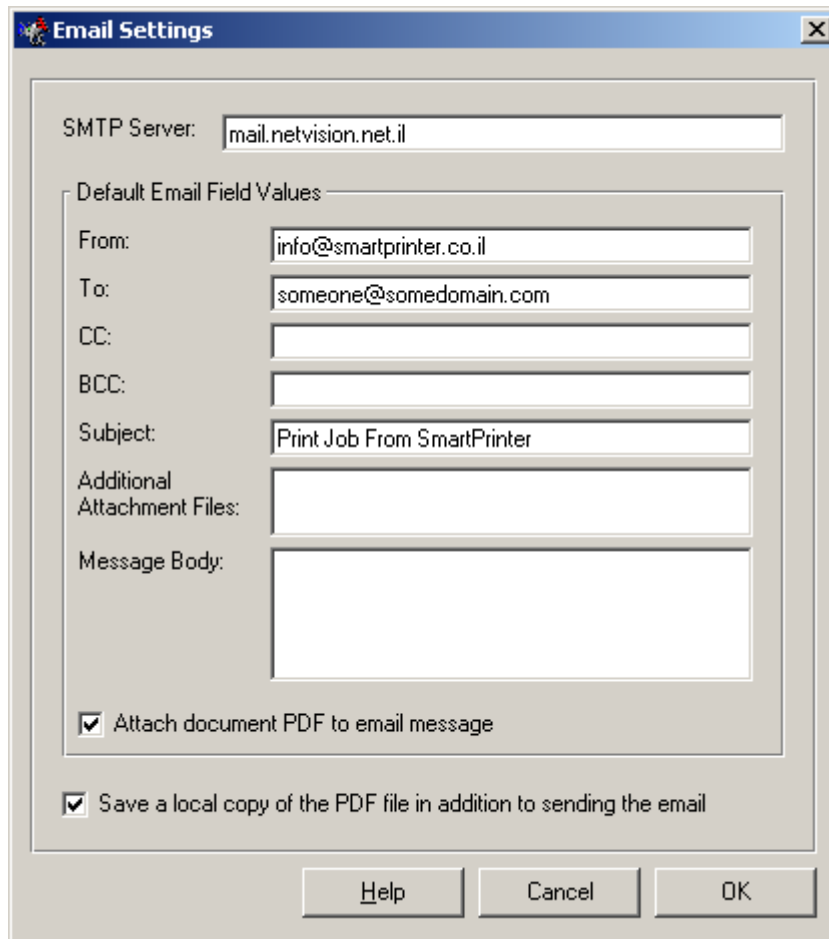


When using a GDI printer, the driver of the SmartPrinter should be the original printer driver, supplied with the printer.



The PDF and TIFF drivers to be used along with the SmartPrinter must support the automatic saving of a file to a folder according to the print job name, without opening a dialog box.

PrintController Email Settings Dialog Box



SMTP Server: mail.netvision.net.il

Default Email Field Values

From: info@smartprinter.co.il

To: someone@somedomain.com

CC:

BCC:

Subject: Print Job From SmartPrinter

Additional Attachment Files:

Message Body:

☒ Attach document PDF to email message

☒ Save a local copy of the PDF file in addition to sending the email

Help Cancel OK

The Email Settings dialog box is used to control all parameters related to sending an email from SmartPrinter. The available options are:

- * **SMTP Server**

The email server must be a valid SMTP server that allows relay of emails from the server on which the SmartPrinter is installed.

- * **From**

The "From" field must be an email address that will not be blocked by the email server and will be allowed to be relayed.

- * **To, CC, BCC**

The "To", "CC" and "BCC" fields are the default values to send any emails that were not specifically changes using the page script.

- * **Subject, Message Body**

The "Subject" and "Message Body" fields are the default values for any emails that were not specifically changes using the page script.

* **Additional Attachment Files**

If you want to attach more files to the email sent from the SmartPrinter server, you need to specify it here. If you want more than one file to be attached, you need to write all file names separated by a semicolon (;) character.

* **Attach document PDF to email message**

If you want to cause SmartPrinter to attach the job file itself to the email as a PDF file, you need to check this checkbox.

* **Save a local copy of the PDF file in addition to sending the email**

If you want to cause SmartPrinter to save the PDF file to a folder in addition to attaching the job file itself to the email as a PDF file, you need to check this checkbox.

PrintController Internal PDF Writer Settings Dialog Box

Internal PDF writer settings

Directory to save PDF files:

☒ Overwrite existing files.
☐ Use unique number as file name
☐ Use job name as the PDF file name

☒ Digitally sign the PDF file

Signature file:
Signature file password:

Author: Producer:
Title: Reason:
Subject: Contact:
Keyword: Location:
Creator: Signed By:

☒ Add a visual signature to the PDF

Signature image file:
Signature position: X: Width:
Y: Height:
Add signature to page:

The Internal PDF Writer Settings dialog box is used to control all the parameters related to saving the print job as a PDF file, these settings also apply to the PDF file attached to the email message in case the job was sent by email.



In order to work with an internal PDF writer, you first must install a Postscript Printer, we recommend using a simple Postscript driver from the ones that are built in the windows system, and not complicated Postscript drivers.

This printer needs to have a local dummy port, you can create such a port by adding a new port, selecting local port, and typing a name of a file such as "c:\temp.ps".



In order to work with an internal PDF writer, you first must install GhostScript. Only after you install GhostScript can SmartPrinter internal PDF Writer create PDF files.

*** Directory to save PDF files**

This will set the root folder to which SmartPrinter will save the created PDF files. If you set the PDF file name to contain backslash characters, you can create the PDF files inside sub folders under the root folder.

*** Overwrite existing files**

When this checkbox is checked, files created by the SmartPrinter will overwrite existing files (if there are such) in the destination folder.

*** Use unique number as file name**

When this option is selected, files created by the SmartPrinter will receive a unique number in the destination folder, but the filename is not under your control.

*** Use job name as the PDF file name**

When this option is selected, files created by the SmartPrinter will receive the same name as the print job from which they were created. Since the job name can be changed using the #SetJobName script command, you can set the file name to be whatever you want using the page script.

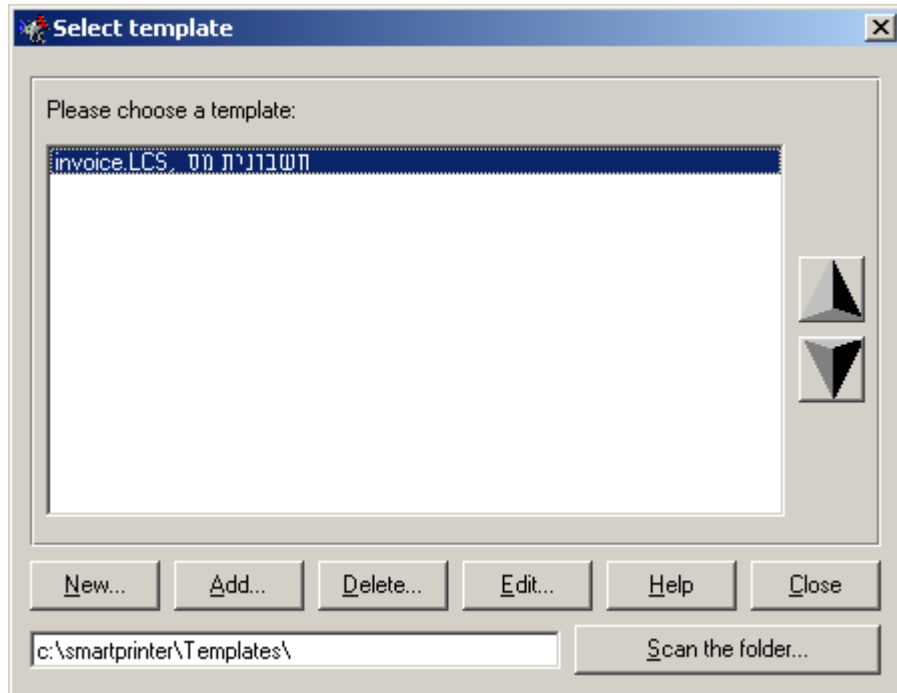
*** Digitally sign the PDF file**

By checking this checkbox and setting the appropriate parameters, you will be able to digitally sign the PDF files created by the SmartPrinter during their creation process. This will allow you to either save these digitally signed files in an archive or send these digitally signed files directly by email to the end customer.

The mandatory parameters to use a digital signature are the Signature file and the signature file password, but it is recommended to add other parameters as well.

Select Template

The select template dialog box enables you to select the template you wish to edit. It also enables you to maintain the list of templates that will be in use by the SmartPrinter in the [template identification phase](#).



Select Template Operations

- * **Up and Down arrows**

These buttons allow you to change the order of the templates in the list. The order of the templates in the list is important since this is the order in which the SmartPrinter is examining a print job to find out whether a template's Key is found. When you click on a template and then click an arrow, the template will move in that direction in the list.

- * **<New...>**

This button is used to create a new template. When you click it, the SmartPrinter will ask you for a name to the file for the template. That template will be automatically added to the templates list.

- * **<Add...>**

This button is used to add an existing template to the templates list. When you click it, the SmartPrinter will ask you for the file of the template to be added, that template will be added to the templates list.

- * **<Delete...>**

This button is used to delete an existing template from the templates list. When you click it, you will be asked whether you like to delete the template from the

list, or erase the file of that template as well.

* **<Edit...>**

This button is used to edit an existing template from the templates list. When you click it, the selected template from the list will be opened, and you will be able to make changes to that template.

* **<Help>**

This button shows this help file.

* **<Scan the folder...>**

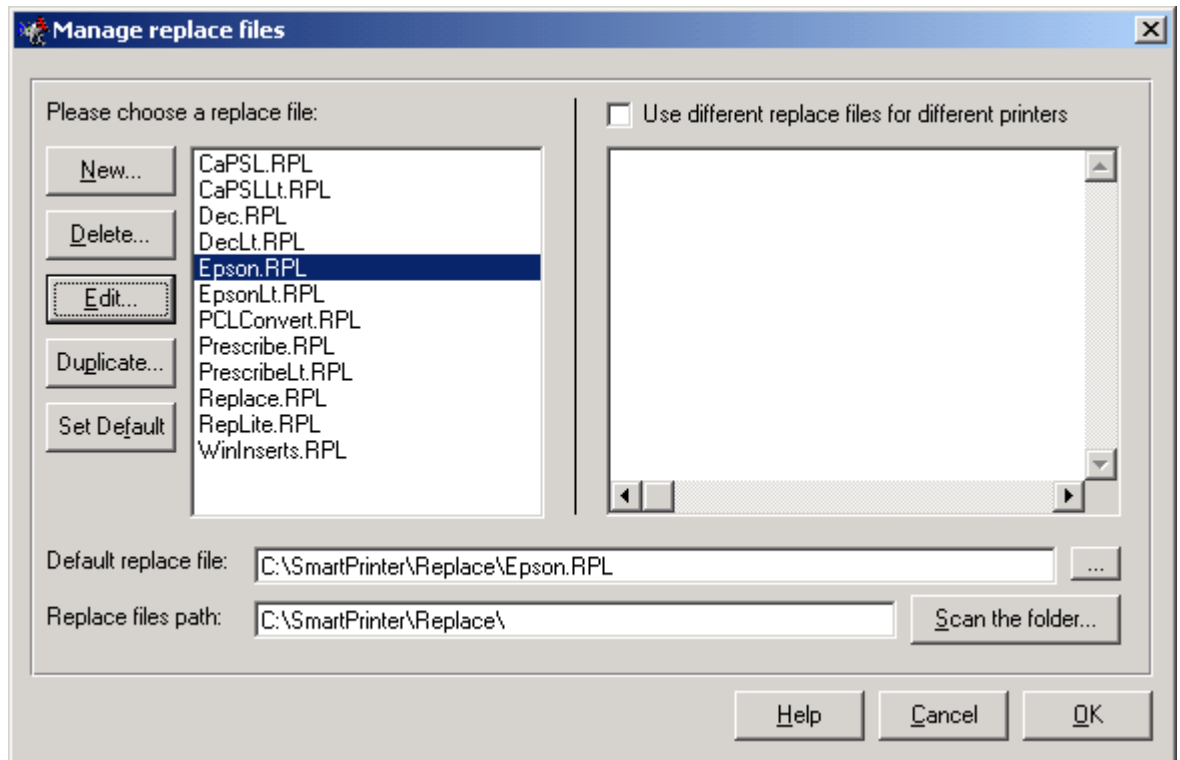
This button is used to create a new list of templates from template files in a folder, or to add a large amount of templates to an existing list. When you click it, you will be asked whether or not you like to erase the existing template list: If you choose <yes> all the existing list of templates will be erased and the new list will be composed only from the new templates that were found during the scan of the folder. If you choose <No> the existing list of templates will remain, and all the templates found during the scan will be added to the end of the existing list.



When scan a folder for templates and don't erase the existing template list, a duplicates of the same template can occur. Make sure to delete any duplications from the list.

Manage Replace Files

The manage replace files dialog box enables you to select the replace file you wish to edit. It also enables you to use a different replace file for each of the licensed printers. The replace files edited here will be in use by the SmartPrinter in the [Replace phase](#).



Manage Replace files Operations

* [<New...>](#)

This button is used to create a new replace file. When you click it, the SmartPrinter will ask you for a name for the replace file. That replace file will be automatically added to the list of replace files.

* [<Delete...>](#)

This button is used to delete an existing replace file from the list.

* [<Edit...>](#)

This button is used to edit an existing replace file from the list. When you click it, the selected replace file will be opened, and you will be able to make changes to that replace file.

* [<Duplicate...>](#)

This button is used to duplicate an existing replace file. It will copy the selected replace file to a new replace file that will contain the same definitions.

* [<Set Default>](#)

When you click this button, the selected replace file from the list will become the default replace file. It will appear in the Default replace file text box.

* **<Help>**

This button shows this help file.

* **<Cancel>**

This button closes the dialog box without saving the changes made.

* **<OK>**

This button saves the changes made and closes the dialog box.

* **Default Replace file**

The replace file appears here is the default replace file that will be used for all printers that no other specific replace file was chosen for it..

* **Replace files path**

This is the path in which all the replace files are stored, the replace files in this path can be seen on the Replace files list on the left. You can click the <Scan the folder...> button in order to refresh the list.

* **Use different replace files for different printers**

When this checkbox is checked, you can edit the table of replace files below it in order to use a specific replace file for a specific printer.

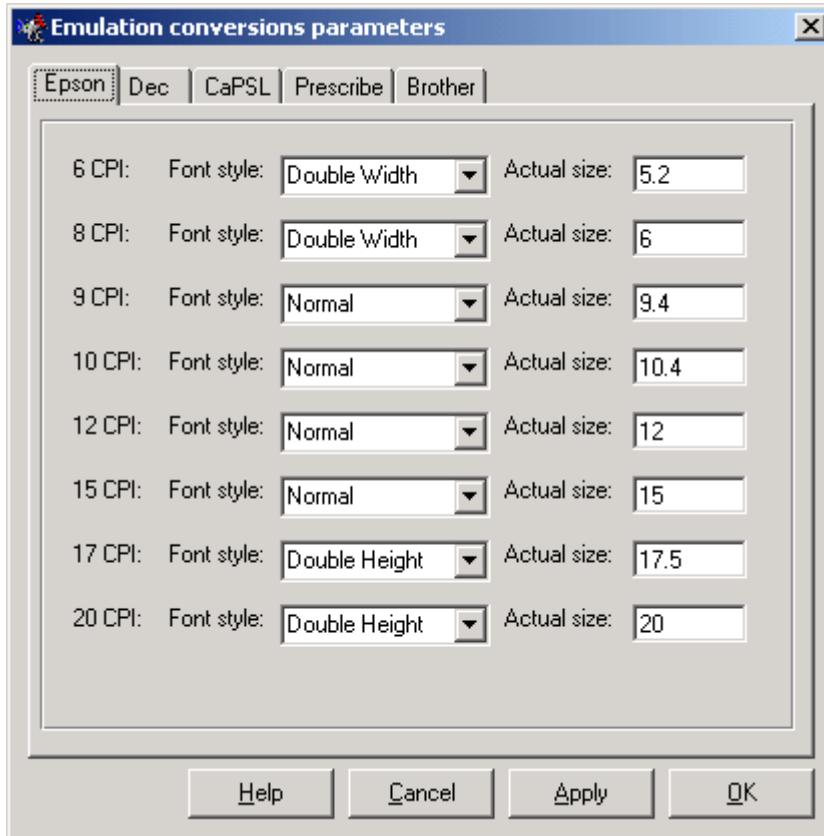
For example: lets say you wish to use the file

“C:\SmartPrinter\Replace\NewReplace.RPL” for the printer “HP LaserJet 4” and the file “C:\SmartPrinter\Replace\Replace.RPL” for all other printers. In this case you write the file “C:\SmartPrinter\Replace\Replace.RPL” in the “Default replace file” textbox, check the “Use different replace files for different printers” checkbox and write:

HP LaserJet 4= C:\SmartPrinter\Replace\NewReplace.RPL
In the textbox below.

PrintController Epson Emulation Conversion Parameters

The Epson emulation conversion parameters dialog box contains all the parameters controls the automatic conversion of print jobs containing Epson ESC/P codes.



CPI	Font style	Actual size
6 CPI	Double Width	5.2
8 CPI	Double Width	6
9 CPI	Normal	9.4
10 CPI	Normal	10.4
12 CPI	Normal	12
15 CPI	Normal	15
17 CPI	Double Height	17.5
20 CPI	Double Height	20

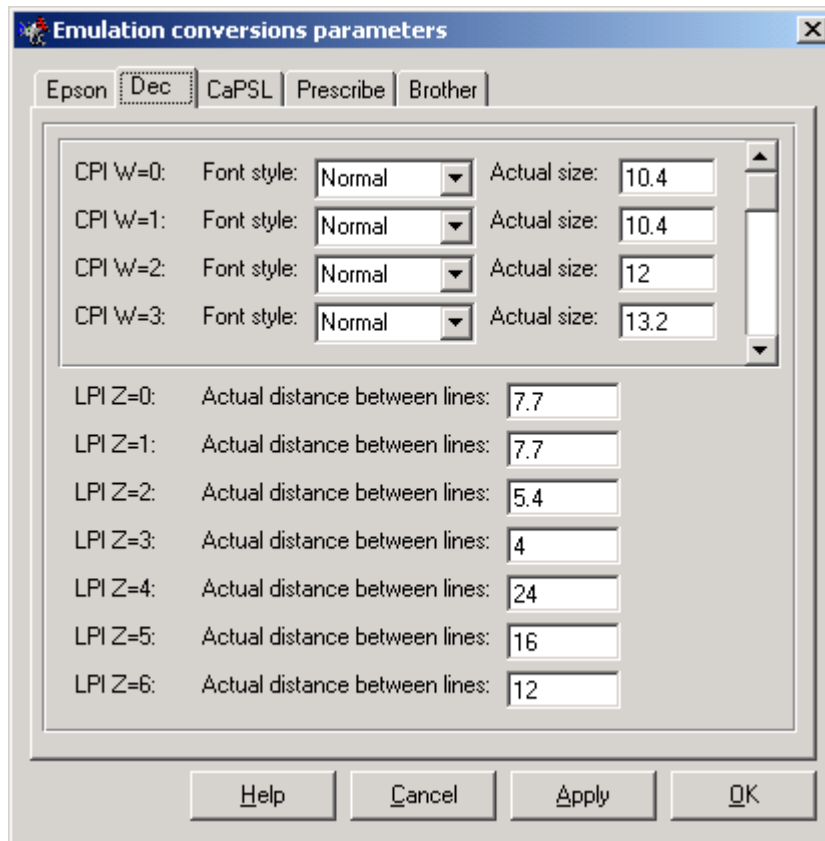
Epson ESC/P basically has 8 sizes of fonts. The SmartPrinter conversion of Epson to PCL codes allow you to decide what will be the real size used as a substitute to the original Epson command. You can choose the real CPI of each font and weather the font shape of each size will be double width or condensed.



SmartPrinter also supports these emulation conversions: [Digital Dec](#), [Canon CaPSL](#), [Kyocera Prescribe](#), [Brother barcodes](#).

PrintController Dec Emulation Conversion Parameters

The Dec emulation conversion parameters dialog box contains all the parameters controls the automatic conversion of print jobs containing Digital DEC codes.



Digital DEC basically has 16 sizes of fonts. The SmartPrinter conversion of Dec to PCL codes allow you to decide what will be the real size used as a substitute to the original Dec command. You can choose the real CPI of each font and weather the font shape of each size will be double width or condensed. In addition the Dec emulation has 7 possibilities for number of lines per inch, you can decide for each option from the Dec command what will the equivalent of the PCL will be.



SmartPrinter also supports these emulation conversions: [Epson ESC/P](#), [Canon CaPSL](#), [Kyocera Prescribe](#), [Brother barcodes](#).

PrintController CaPSL Emulation Conversion Parameters

The CaPSL emulation conversion parameters dialog box contains all the parameters controls the automatic conversion of print jobs containing Canon CaPSL codes.

The screenshot shows a Windows-style dialog box titled "Emulation conversions parameters". It has five tabs: "Epson", "Dec", "CaPSL" (which is selected), "Prescribe", and "Brother". The "CaPSL" tab contains the following settings:

- 10 CPI: Actual size:
- 12 CPI: Actual size:
- 15 CPI: Actual size:
- Double height/width factor:

- 3 LPI: Actual distance between lines:
- 4 LPI: Actual distance between lines:
- 6 LPI: Actual distance between lines:
- 8 LPI: Actual distance between lines:
- 12 LPI: Actual distance between lines:

Background shading fill: %

At the bottom of the dialog are four buttons: "Help", "Cancel", "Apply", and "OK".

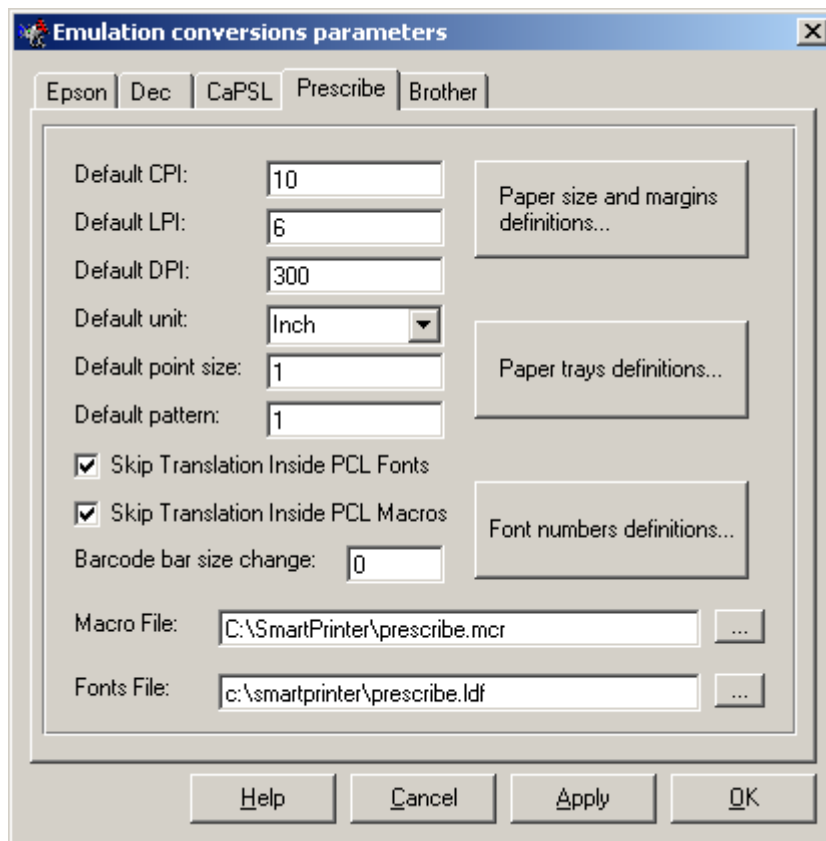
CaPSL basically has 3 sizes of fonts. The SmartPrinter conversion of CaPSL to PCL codes allow you to decide what will be the real size used as a substitute to the original CaPSL command. You can choose the real CPI of each font. Additionally the CaPSL allow you to set double width and/or height for each font size. The “Double height/width factor” parameter sets the factor in which the font size will be multiplied by. Additionally the CaPSL emulation has 5 possibilities for number of lines per inch, you can decide for each option from the CaPSL command what will the equivalent of the PCL will be. You can also select the background shading grayscale amount that will be used to fill boxes.



SmartPrinter also supports these emulation conversions: [Epson ESC/P](#), [Digital Dec](#), [Kyocera Prescribe](#), [Brother barcodes](#).

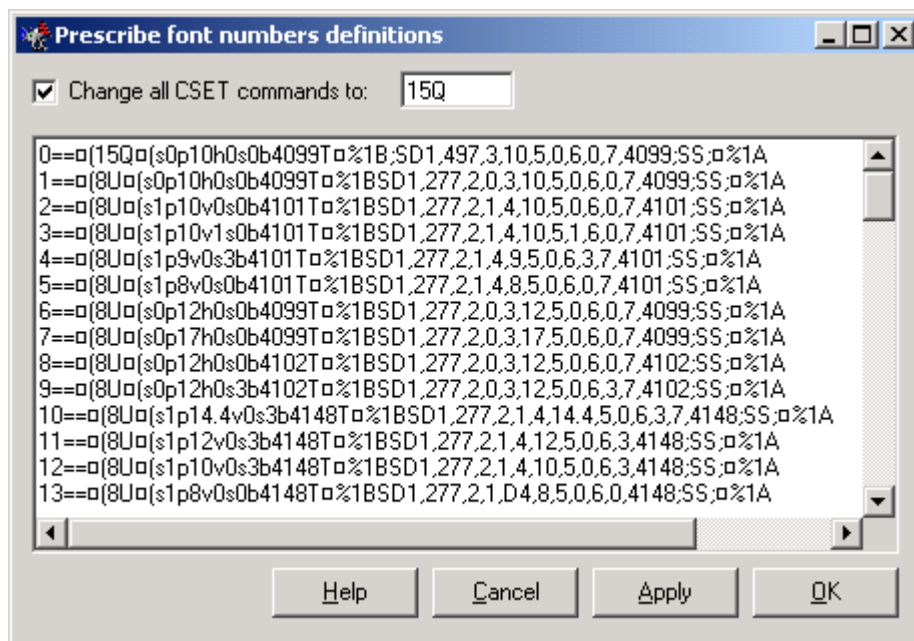
PrintController Prescribe Emulation Conversion Parameters

The Prescribe emulation conversion parameters dialog box contains all the parameters controls the automatic conversion of print jobs containing Prescribe codes.

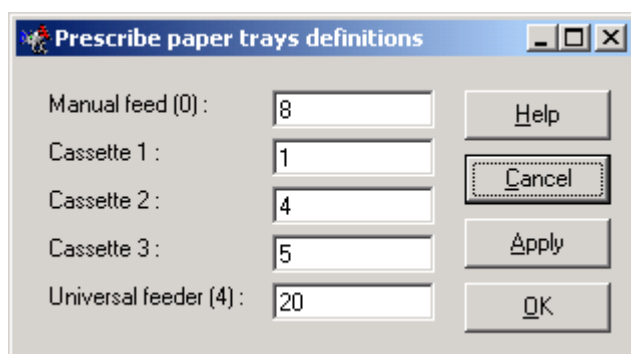


Here you can set all the initial values (such as CPI, LPI, DPI, Unit, etc...) for the prescribe environment.

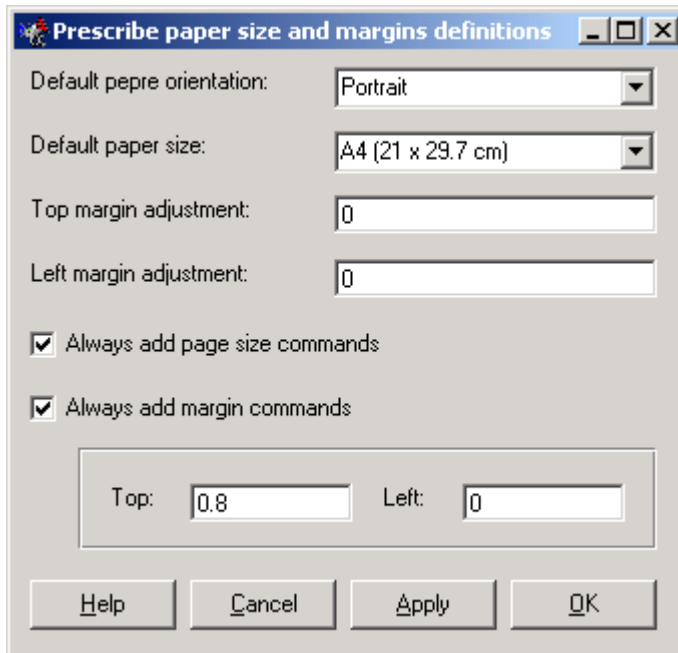
The prescribe language has the ability to load macro files to the printer and then call it while printing. Since SmartPrinter is designed to work on any printer supports PCL5 language, regardless of whether it is capable of storing macros, we put all relevant macros in a file, and when there is a need to load a macro, the SmartPrinter will search for it in this file, and use it.



The Prescribe language has many fonts, and there is also an option to load more fonts and define fonts while printing. In the font numbers definitions textbox you can define for each prescribe font number the equivalent PCL font selection command.



The Prescribe language is capable of printing from several paper trays, here you can define each prescribe paper tray to a PCL paper tray code according to your printer's paper tray selection codes.



Prescribe paper size and margins definitions

Default paper orientation: Portrait

Default paper size: A4 (21 x 29.7 cm)

Top margin adjustment: 0

Left margin adjustment: 0

☒ Always add page size commands

☒ Always add margin commands

Top: 0.8 Left: 0

Help Cancel Apply OK

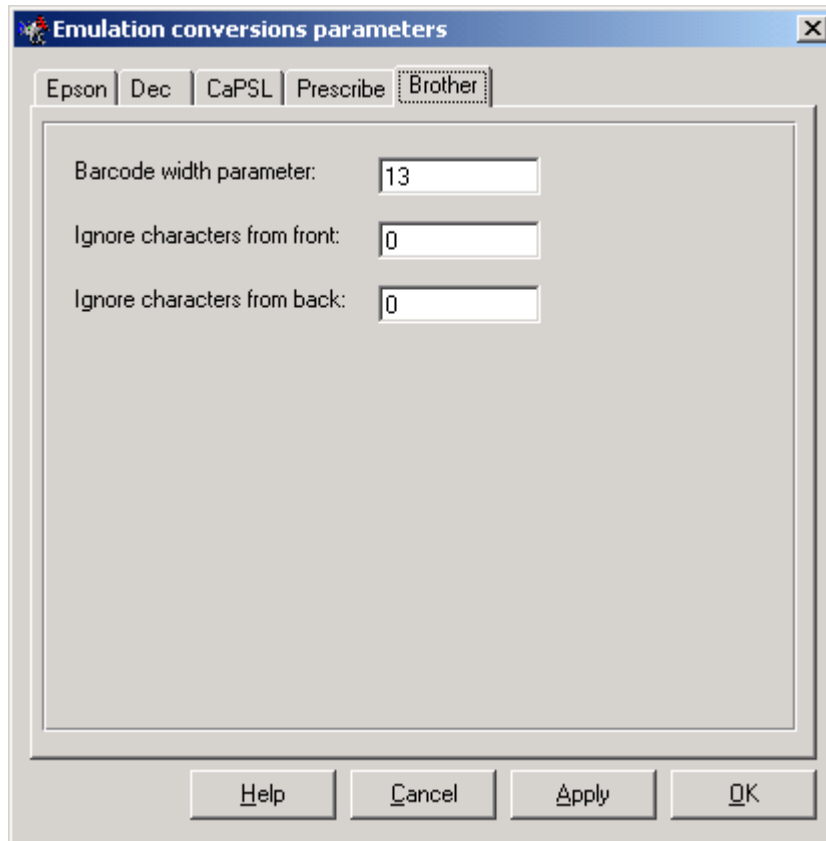
The Prescribe language is capable of printing on several paper sizes, and to change the margins and orientation, here you can select the default paper size and orientation, as well as make adjustments to the margins and/or make your own margins.



SmartPrinter also supports these emulation conversions: [Epson ESC/P](#), [Digital Dec](#), [Canon CaPSL](#), [Brother barcodes](#).

PrintController Brother Barcode Emulation Conversion Parameters

The Brother barcode emulation conversion parameters dialog box contains all the parameters controls the automatic conversion of print jobs containing Brother barcode escape codes.



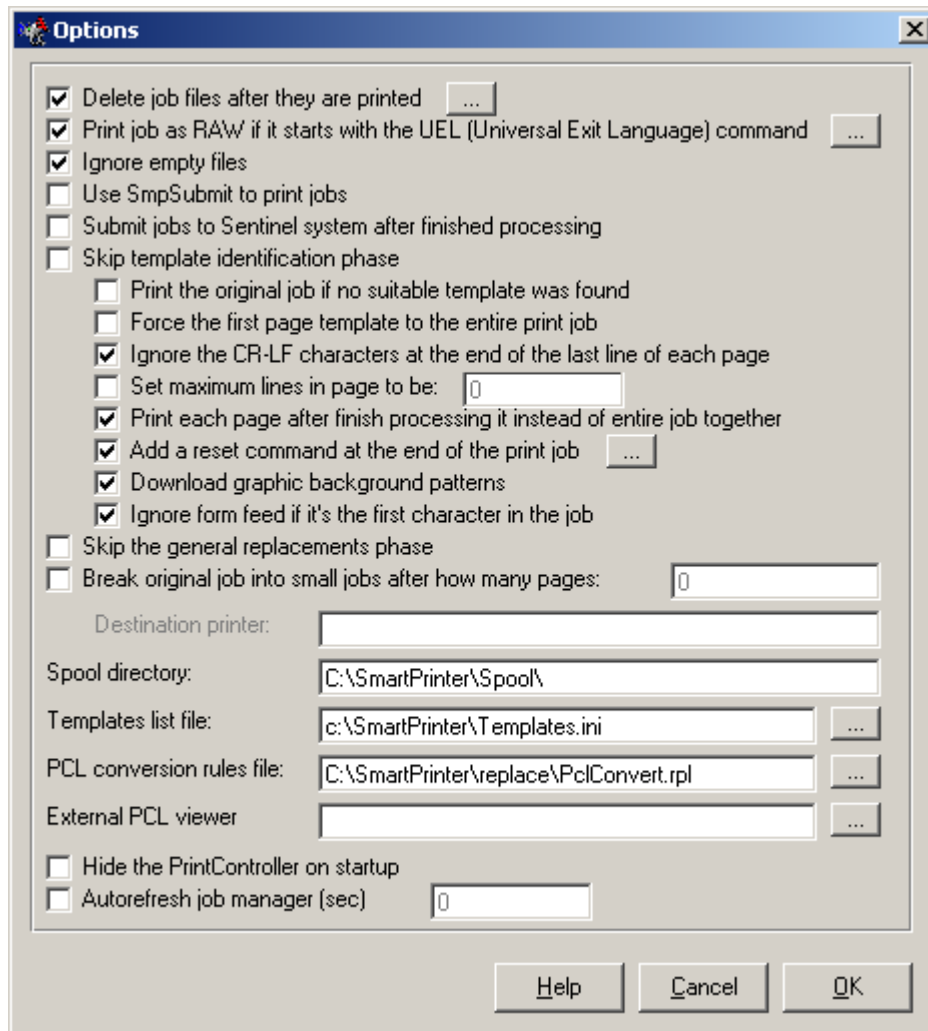
Brother barcodes are activated by a special customized escape code, unique for brother printers, the SmartPrinter is able to automatically understand the barcode commands and translate it to simple standard PCL commands.



SmartPrinter also supports these emulation conversions: [Epson ESC/P](#), [Digital Dec](#), [Canon CaPSL](#), [Kyocera Prescribe](#).

PrintController Options Dialog Box

The options dialog box contains all kind of parameters that controls the different possibilities of the SmartPrinter functionality.



The available options are:

- * **Delete job files after print**

When this checkbox is checked, some of the temporary files created during the job manipulation process are deleted after the manipulated job was printed. When this checkbox is unchecked, these temporary files are not deleted.

In order to select which temporary files will be deleted and which will not be deleted, click the <...> button, which will activate the [delete temporary files dialog box](#).

This checkbox controls the same functionality as the File | Save print jobs menu.

- * **Print jobs as RAW if it starts with the UEL command**

When this checkbox is checked, any print job that starts with the UEL (Universal Exit Language) command will be sent as is (RAW) to the printer and will not be handled.

Usually, graphic print jobs start with that command, and then the SmartPrinter will know

to ignore these print jobs.

However, some drivers create a different command than the standard UEL command, in these cases, you can edit the UEL command by clicking the <...> button and [edit the UEL Command](#) Sequence. Click [here](#) for more information on the way to edit the sequence.

*** Ignore empty files**

When this checkbox is checked, the SmartPrinter will ignore all incoming files that are empty, and will not run them through the Replace phase and the Template identification phase. However, the Last job number parameter will advance when an empty file arrives.

*** Use SmpSubmit to print jobs**

When this checkbox is checked, the SmartPrinter will use a special method called SmpSubmit to print the jobs. The meaning is that each print job is calculated within a different process. The advantage is that even if one print job is bad and crushes the process, the rest of the print jobs are not harmed. The disadvantage is that the order in which the print jobs arrived to the SmartPrinter will not necessarily be kept.

Use this function only if the order of the print jobs is not important. Usually being used when there are many small jobs with many scattered printers.

*** Submit jobs to Sentinel system after finished processing**

When this checkbox is checked, the SmartPrinter will process to print jobs as usual, but after finished processing, it will not print the finished jobs to the printer, but it will submit the processed jobs into the Sentinel system. The Sentinel system will take control over the rest of the workflow until the time these jobs will be printed. To read more about the Sentinel system you can visit our web site at: <http://www.smartprinter.co.il/en/sentinel.html>

*** Skip template identification phase**

When this checkbox is checked, the template identification phase is not being preformed for the arriving print jobs. The template identification phase is a relatively heavy step, and may slow the print process when the print jobs are large.

Please refer to [SmartPrinter Process Procedure](#) for more information regarding the print process.

*** Print the original job if no suitable template was found**

When this checkbox is checked, the SmartPrinter will process the print job as usual, but after the template identification phase, if none of the pages of the print job was identified by any template, the SmartPrinter will print instead of the processed file, the original file received (the file before the general replacements phase).

*** Force the first page template to entire job**

When this checkbox is checked, and the first page of the print job was positively identified with template, all the rest of the print job pages will be forced to be positively identified by that same template, even if they don't have the matching keys for that template.

If the first page of the job was not positively identified with any template, that parameter has no effect.

*** Ignore the CR-LF characters at the end of the last line of each page**

This parameter handles only the template identification phase and not the general replacements phase.

When this checkbox is checked, the characters CR-LF, that represents the end of line for the last line of a page will not be written, instead only a FF character will be written.

*** Maximum lines in page**

This parameter handles only the template identification phase and not the general replacements phase.

When this checkbox is checked, the SmartPrinter will still look for the FF character to know where the pages ends, but if it will not find this character after X lines it will act as if it found the FF character. X is the parameter written in the textbox next to the checkbox.

*** Print each page after finish processing it instead of entire job together**

This parameter handles only the template identification phase and not the general replacements phase.

When this checkbox is checked, the SmartPrinter will read a page, process it and then send it to the output file. The result is that no changes can be made on one page according to data from another page. On the other hand, the process is much faster and takes less memory, since there is no need to hold the entire job in memory.

*** Add a reset command at the end of the print job**

This parameter handles only the template identification phase and not the general replacements phase.

When this checkbox is checked, the SmartPrinter will a reset command to the end of the print job, after the EndJob parameter. This is used since some printer fail to eject the last page until they receive the reset command. You can [edit the reset command](#) that will be edited according to the type and model of your printer, by clicking the <...> button.

*** Download graphic background patterns**

This parameter handles only the template identification phase and not the general replacements phase.

When this checkbox is checked, the SmartPrinter will a download the graphic backgrounds for the page scripts that handles graphics (such as #DrawFullBox and #DrawFullCircle).

*** Ignore form feed if it's the first character in the job**

This parameter handles only the template identification phase and not the general replacements phase.

When this checkbox is checked, the SmartPrinter will a check if the print job starts with a FormFeed character, and ignores it if so (it was common to send a FormFeed character at the beginning of the print job in line printers, to advance the printer to the beginning of the page before the printer starts printing).

*** Skip general replacements phase**

When this checkbox is checked, the general replacements phase is not being preformed. Please refer to [SmartPrinter Process Procedure](#) for more information regarding the print

process.

* **Break original job into small jobs after how many pages**

When this checkbox is checked, the SmartPrinter PrintProcessor will process the print job in parts. The purpose of this feature is to allow the printer to start working on the beginning of a long print job, even if not all the print job was sent to the printer.

* **Number of pages**

This parameter sets the number of pages after which the job will be break and the process on the received part will start. If the parameter is set to 0 (default) then the entire job is processed together.

* **Destination Printer**

This parameter enables to send the divided and manipulated print jobs to a different printer in order to not disturb the receive process of the rest of the print job.

* **Spool Directory**

The spool directory is the folder in which the temporary job files are stored.

* **Template List File**

The template list file is the file contains the list of all templates to be searched for during the [template identification phase](#). The list also contains the files in which the template rules are stored.

* **PCL conversion rules file**

The PCL conversion rules file is the file contains all the rules to be used during the conversion of a print file to a SmartPrinter file in the [SmartPrinter file converter dialog box](#).

* **External PCL viewer**

Here you can select an external PCL viewer, after you do that, the “view job” option will be enabled in your job and file menus.

* **Hide the PrintController on startup**

When this checkbox is checked, the PrintController will be started as hidden, when only an icon on the systray is an indicator that the Controller is running.

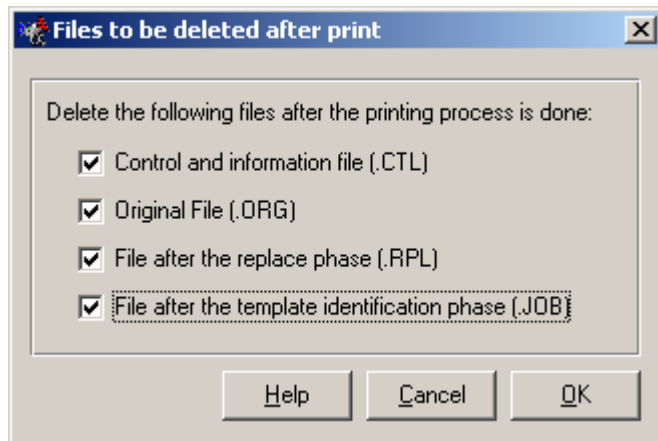


* **Auto refresh job manager**

The auto refresh job manager option allow you to instruct the PrintController to refresh the list of saved jobs every number of seconds.

PrintController Delete temporary files

The delete temporary files dialog box is the place where you can select which file types will be deleted after the print job is preformed, and which will be kept.



Simply select the types of files that you wish that they will be deleted, and unselect the files you wish to keep, and then click the <OK> button.

The types of temporary files the SmartPrinter may produce are:

- *.ORG These are the original files as came from the sending application without any changes.
- *.CTL These files are control files containing information about the received files. Information such as job name, user who sent the job, time and date, etc...
- *.RPL These are the files created after the Replace phase.
- *.JOB These are the files created after the Template phase, these are the final files being send to the printer for actual print.

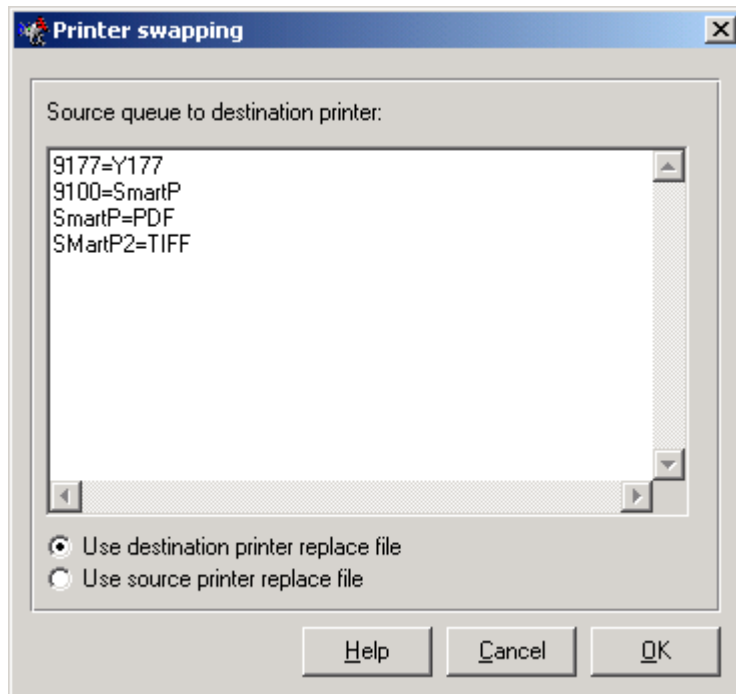


The job manager display of jobs is based on the control files (*.CTL), which means that if the control file is saved the job will be displayed on the job manager, and if not it will not show there.



If the replace phase is skipped, then no *.RPL files are created, and there is no need to delete it.

PrintController Printer Swapping Dialog Box



The printer swapping dialog box is used to transfer print jobs arriving from one printer/queue/port to another printer, this is used for several purposes:

1. When using the SmartPrinter internal LPD server, use this table to make the connection between the name of the remote printer (queue) as defined in the host, and the name of the windows printer to which the print job should eventually be printed.
For example, if in the host there is a queue that its remote printer name is defined to be "PRT01", and in the windows on which the SmartPrinter is installed there is a printer called "HP LaserJet 4", and you want any print job that arrives from this queue will be printed to that printer, you should write on the table:
PRT01=HP LaserJet 4
2. When using the SmartPrinter internal RAW server, use this table to make the connection between the number of the port as defined in the host, and the name of the windows printer to which the print job should eventually be printed.
For example, if in the host sends the print job to the port 9100, and in the windows on which the SmartPrinter is installed there is a printer called "HP LaserJet 4", and you want any print job that arrives from this port will be printed to that printer, you should write on the table:
9100=HP LaserJet 4
3. When from some reason you want that a job arriving one printer will be printed on a different printer.

4. When you want to define a fax printer, you should write on the table:
SmartPrinter=FAX
This will define that any printer arriving the "SmartPrinter" printer will not be printed, but sent via fax.
5. When you want to define a pdf/tiff printer, you should write on the table:
SmartPrinter=PDF
This will define that any printer arriving the "SmartPrinter" printer will not be printed, but sent to a PDF Writer or to a TIFF Driver to be saved as a file.
6. When you want to define a printer as a GDI (NON PCL) printer, you should write on the table:
SmartPrinter=GDI
This will define that the "SmartPrinter" printer is not a PCL printer, and then when the print job is finished processing by the SmartPrinter, it will be converted using the real printer driver to a GDI output suiting the physical printer.



In order to send a print job via fax, you need to install and configure "Microsoft Fax" on your SmartPrinter Server computer. Moreover, SmartPrinter supports this feature only in Windows NT/2000/XP/2003 and not in Windows 95/98/ME



In order to send a print job via fax, you need to have the SmartPrinter Fax module licensed.



In order save a print job to PDF/TIFF file, you need to install and configure PDF Writer of TIFF Driver, these drivers should support an automatic save of a file to a folder according to the name of the print job.



In order save a print job to PDF/TIFF file, you need to have the SmartPrinter PDF module licensed.

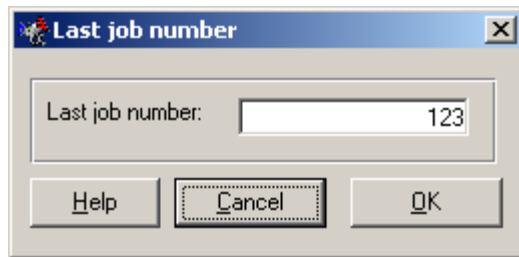


If you are using [different replace files for different printers](#) select from the which replace file will be used, the source or the destination printer replace file.



In order print the job directly to a port such as COM1 or LPT1, just write the printer name = port name with a colon at the end, such as:
SmartP=COM1:

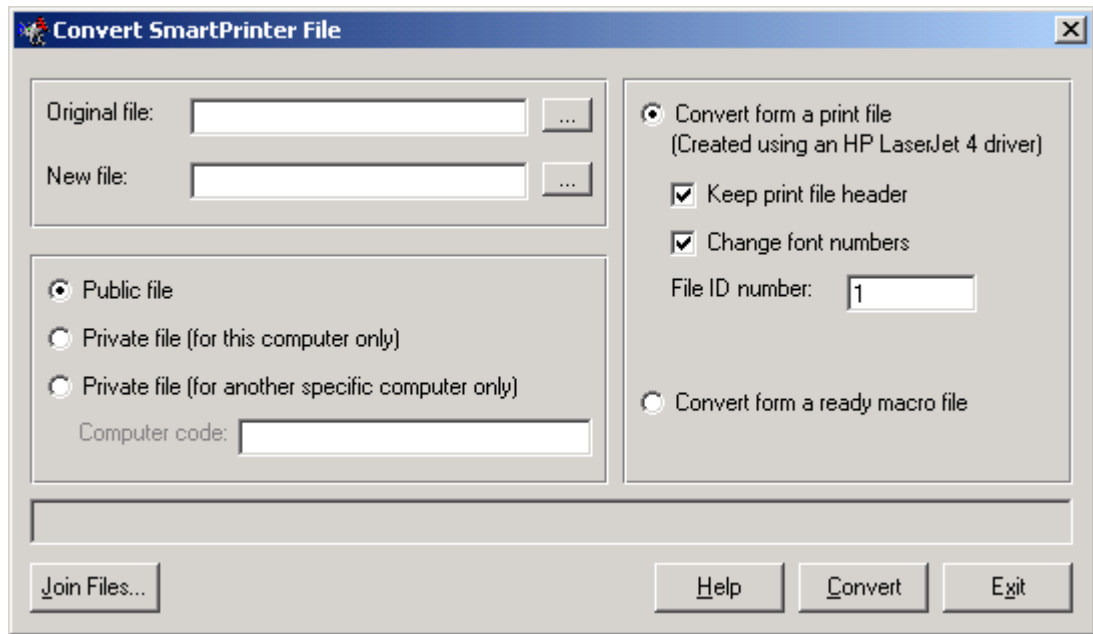
PrintController Last Job Number Dialog Box



The last job number dialog box allow you to change the number of the last job entered the SmartPrinter system, The last job number is the number of the file for the last job that was already written and processed by the SmartPrinter. It means that the next file to be written in the following number.

PrintController Convert SmartPrinter Files Dialog Box

The Convert SmartPrinter file dialog box is the place where you can convert macro files or output print files into SmartPrinter Files (SPF).



A SmartPrinter file is a unique file type suitable for use with SmartPrinter. Though it is possible to use any binary file with SmartPrinter, SPF has 2 major advantages:

1. It is encoded. Any attempt to send the file directly to the printer without going through the SmartPrinter mechanism will fail.
2. It can be either public or private.
 - * A **public** SPF can be used in any computer running SmartPrinter.
 - * A **private** SPF can be used only at a specific computer, any attempt to use it on a different computer other than the one it was prepared for will fail.

In order to convert a file to SPF, follow these steps:

1. Type the source file (or select it using the <...> button) at the “Original File” textbox.
2. Type the destination file (or select it using the <...> button) at the “New File” textbox.
3. Select whether the destination SPF will be public, private (for this computer only) or private for another computer, if you want it to be

private for another computer, you must type the [license code](#) of this computer under the “Computer Code” textbox.

4. Select whether the original file is a macro file or a printer file. If it is a printer file choose a number for the destination file (between 1 and 32767), and choose whether or not to leave the header of the file in the SPF.
5. Click <Convert>.

If you want to join two SmartPrinter files into one, click the <Join Files...> button, this will lead you to the [Join SmartPrinter Files Dialog Box](#)



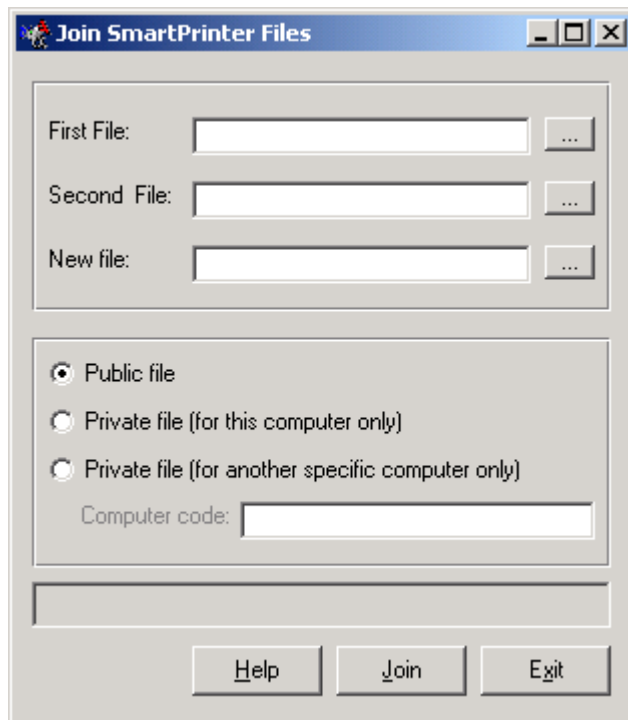
The SmartPrinter converter is designed to work with print files made with the “HP LaserJet 4” windows driver only, any attempt to use it with a different type of file may have unexpected results.



If you are using the converter to create templates for the SmartPrinter WinInsert version, you need to remove the print file header.

PrintController Join SmartPrinter Files Dialog Box

The Join SmartPrinter files dialog box is the place where you can join two [SmartPrinter Files](#) (SPF's) into one file. Since a SmartPrinter file is a coded file format, you can't simply join two SmartPrinter files using a standard utility or a hex editor. If you want to join two SmartPrinter files to one file you must use this dialog box.



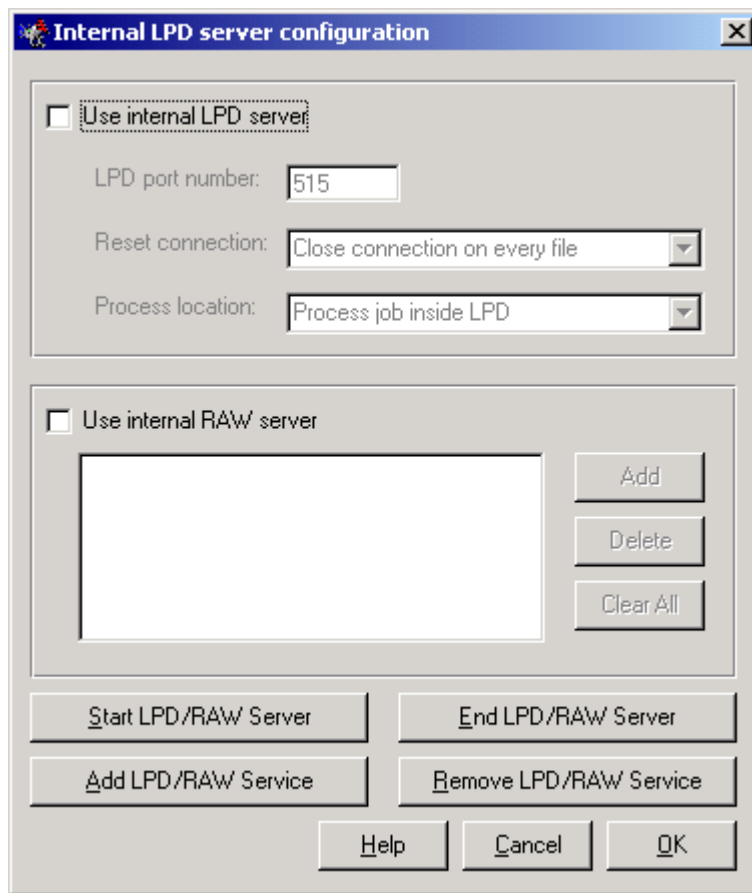
In order to join two SmartPrinter files to one, follow these steps:

1. Type the first source file (or select it using the <...> button) at the "First File" textbox.
2. Type the second source file (or select it using the <...> button) at the "Second File" textbox.
3. Type the destination file (or select it using the <...> button) at the "New File" textbox.
4. Select whether the source SmartPrinter files are public, private (for this computer only) or private for another computer, if they are private for another computer, you must type the [license code](#) of this computer under the "Computer Code" textbox.
5. Click <Join>.



Both source SmartPrinter files must be of the same type, e.g. they both must be public, or both private of the same computer. You can't join a public file with a private file or two private files that were made for different computers.

PrintController Internal LPD Dialog Box



The Internal LPD dialog box contains all parameters that control the internal LPD and RAW servers.

In order to control the translation of LPR Queue name to Printer name in the LPD server, or the translation of RAW Port number to Printer name in the RAW server, please refer to the [Printer Swapping dialog box](#).

LPD Server Options

* Use Internal LPD Server

When this checkbox is checked, the SmartPrinter LPD Server is enabled. You can start it by clicking the <Start LPD/RAW Server> button, or by running the SMPLPD.EXE file.

* LPD Port Number

This parameter sets the port number on which the LPD Server will Listen. According to the RFC1179 protocol, this port must be 515.

* Reset Connection

The LPD Server can run in one of two modes regarding the connection to the host that is sending the print job.

1. Close connection on every file

1. When this option is selected, the LPD Server will close the port and the socket after every file that was accepted from the host.
2. **Keep open connection for all files**
When this option is selected, the LPD Server will keep the port and the socket open even after a file was accepted from the host, and only close the connection and then wait for the next file.

*** Process Location**

The LPD Server can run in one of two modes regarding the place to process the print job.

1. **Process job inside LPD**
When this option is selected, after the LPD Server had accepted a file from the host, it will process it according to the [SmartPrinter process procedure](#), and then send it to the destination windows printer when it is already manipulated. Therefore the destination printer must be a regular printer so that the print job will not be processed twice.
2. **Process job on destination printer**
When this option is selected, the LPD Server will not process data from the host, but send it straight to the destination windows printer (the data will not even be saved to a temporary file). If this file is to be manipulated, then the destination printer should have a SmartPrinter PrintProcessor, and then that printer will make the manipulation on the print job.

RAW Server Options

*** Use Internal RAW Server**

When this checkbox is checked, the SmartPrinter RAW Server is enabled, You can start it by clicking the <Start LPD/RAW Server> button, or by running the SMPLPD.EXE file.

*** Add RAW Port Number**

Using this button, you can add ports that the SmartPrinter RAW Server will listen to.

*** Delete Raw Port Number**

Using this button, you can delete ports that the SmartPrinter RAW Server no longer needs to listen to.

*** Clear All Ports**

Clicking this button will delete all of the ports from the list.

*** <Start RAW/LPD Server>**

Clicking this button will activate the SmartPrinter RAW/LPD Server.

*** <End RAW/LPD Server>**

This button searches whether the SmartPrinter RAW/LPD Server is activated, and if so it closes the LPD Server.



The Internal LPD and Internal RAW servers cannot be used at the same time.

* **<Add RAW/LPD Service>**

Clicking this button will add the SmartPrinter RAW/LPD Server as a service to the Windows Services. This will enable the RAW/LPD Server to start working without the need to log in to the computer.

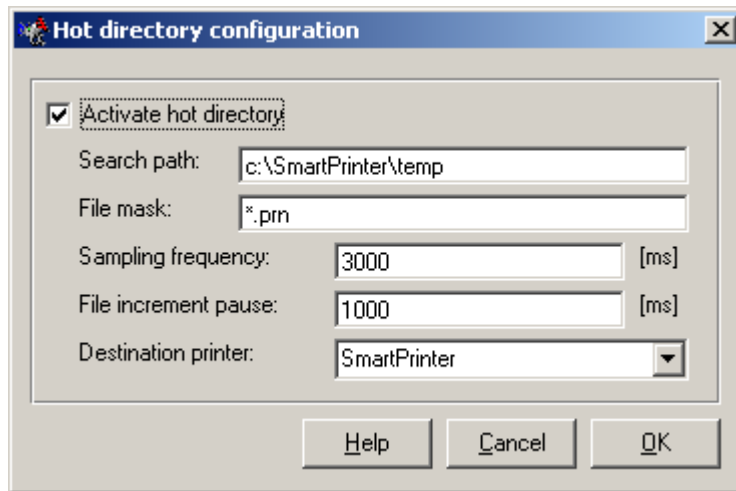
* **<Remove RAW/LPD Service>**

Clicking this button will remove the SmartPrinter RAW/LPD Server from the Windows Services.



The RAW/ LPD Server can be added as a service only on Windows NT and above operating systems, and not on windows 95/98/ME.

PrintController Hot Directory Dialog Box



The hot directory dialog box contains all parameters that control this way to receive jobs into SmartPrinter, click [here](#) for more information on other ways to receive print jobs into SmartPrinter

- * **Activate Hot Directory**

When this checkbox is checked, the SmartPrinter will start scanning for new files in the selected folder.

- * **Sampling Frequency**

This parameter sets the frequency that the folder will be scanned (in 1/1000 second units).

- * **File increment pause**

This parameter sets the interval between tests for the file size (in 1/1000 second units). When a file is discovered, the PrintController tests it's size, then wait this time period and tests the size again. If the sizes are the same, that means that the file wasn't incremented and it is being sent to the destination printer.

- * **Search Path**

This parameter sets where the SmartPrinter will search for new files.

- * **File Mask**

This parameter sets which file will the SmartPrinter search for.

- * **Destination Printer**

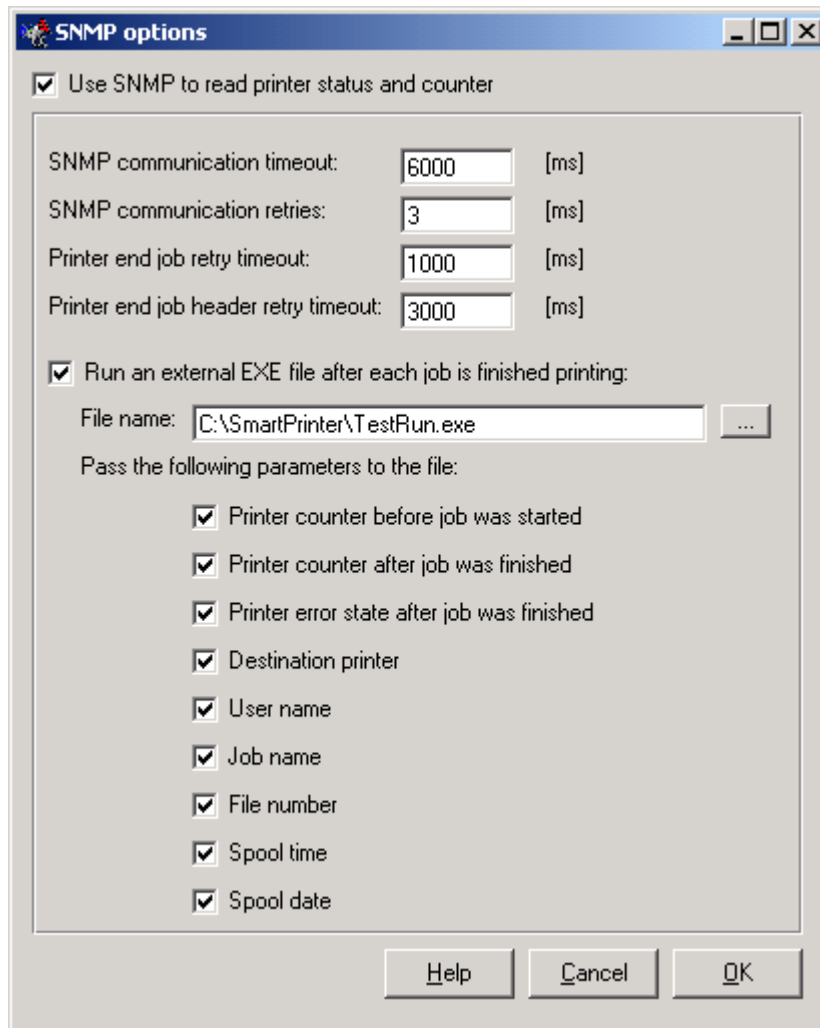
This parameter sets the printer to which the files that were found will be printed.



This way of receiving print jobs into SmartPrinter is unrecommended, aspecially in network applications, since if two or more users print at the exact same time, their print jobs may be written one on top of the other,

and only one print job will be printed!

PrintController SNMP Options Dialog Box



The SNMP options dialog box contains all parameters that control SNMP (Simple Network Management Protocol) usage with SmartPrinter. This is useful if you want to query the printer for the page counter and statuses before/after each job printer to a printer.

- * **Use SNMP to read printer status and counter**

When this checkbox is checked, the SmartPrinter will enable the SNMP option.

- * **SNMP communication timeout**

This parameter sets the timeout for the SNMP query sent to the printer (in 1/1000 second units) the slower and bigger your network is, the bigger this parameter should be.

- * **SNMP communication timeout**

This parameter sets the retries for the SNMP query sent to the printer.

* **Printer end job retry timeout:**

This parameter sets the interval (in 1/1000 second units) between tests sent to the printer to check whether or not the job was finished printing and the printer counter can be read.

* **Printer end job header retry timeout:**

This parameter sets the interval (in 1/1000 second units) the SmartPrinter will wait between the time it received a ready status from the printer, and the time it will send another query to the printer to check whether or not the job was really finished printing or the previous ready status was just a temporary status in the middle of a print job. It is not recommended to set this parameter to less than 3000 ms.

* **Run an external EXE file after each job is finished printing**

This parameter allows you to run an external EXE file after each job that was finished printing.

* **File name**

This parameter sets which file will the SmartPrinter run.

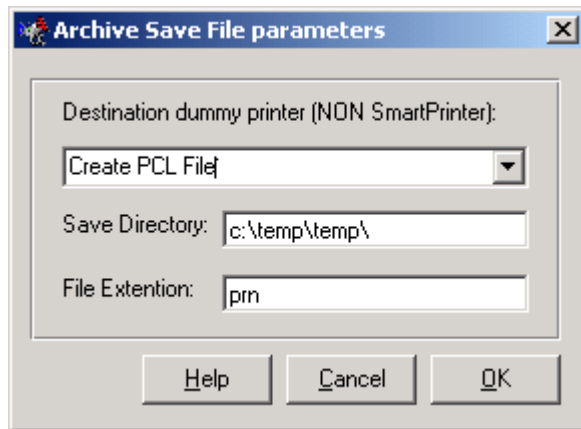
* **Pass the following parameters to the file**

This parameter sets which parameters will be passed to the EXE file that will be executed. Only the parameters that are marked will be passed, and it will be passed at the order they are marked from top to bottom.



The SNMP function will only work with printers support SNMP that are connected via a TCP/IP network connection.

PrintController Archive SaveFile Options Dialog Box



The Archive SaveFile options dialog box contains all parameters that control saving raw data to a file, meaning saving the final job file to an archive directory. These files can be easily reprinted, simply by sending it to a printer, since it is final print job files.

- * **Destination dummy printer**

This is just any dummy printer used by SmartPrinter in order to create the job file. The only condition on this printer is that the print processor of this printer must be WinPrint and not SmartPrinter.

- * **Save directory**

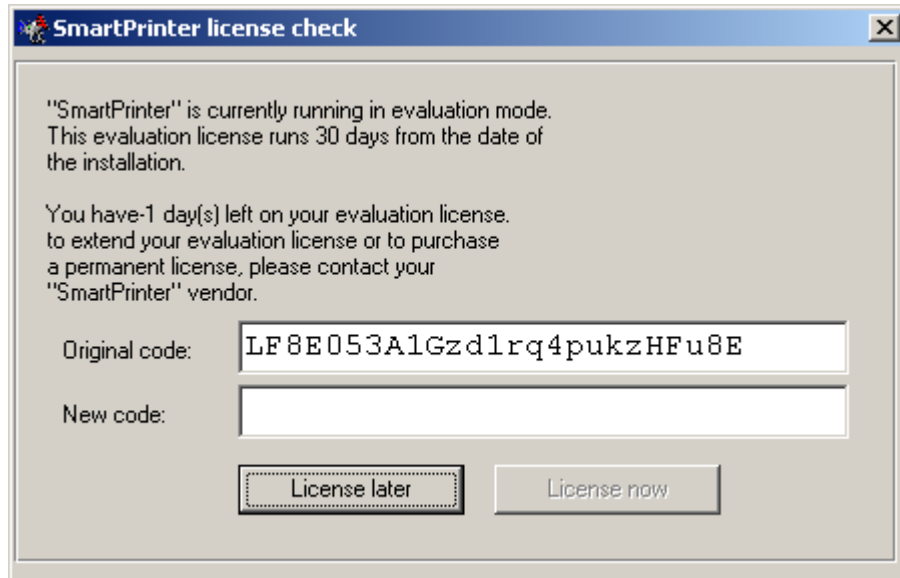
This parameter sets where the job files will be saved, the name of the file is the job name.

- * **File extension**

This parameter sets the extension of the print file that will be saved.

License Dialog Box

The license dialog box is the place where you can change your license code. A new license code can include a permanent license code, a code which allow you to use more printers or any other license code.

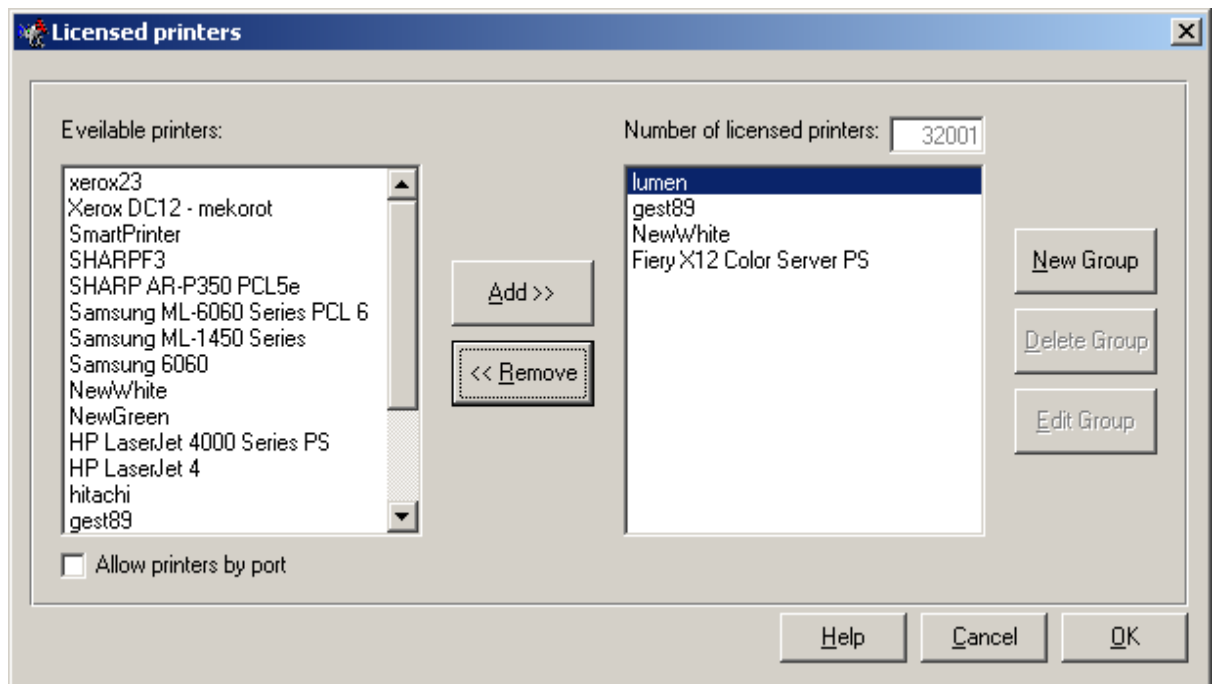


To change the license code to a different license code, you need to give the original code to your SmartPrinter vendor, and he will supply you with a new license code. Enter the new license code to the new code text box, and then click <License Now> the new license code will be processed by the SmartPrinter.



The SmartPrinter license code is computer dependent, it means that you shouldn't try to transfer the license code from one computer to the other since it will not work, and may result in devastation of the SmartPrinter in the new computer.

PrintController Licensed Printers Dialog Box



In the Licensed Printers dialog box, you can select which printers will be the currently licensed printers. The printers listed on the left list are printers that exist in the windows printers list. The printers listed on the right list are the currently licensed printers. The SmartPrinter will allow you to print manipulated print jobs to any licensed printer.

The number of licensed printers parameter shows you how many printers you can add to the licensed printers list.

Additionally, this dialog box is the place where you can manage printer groups.

Licensed Printers Operations

* <Add>

To add a printer to the licensed printers list, you simply need to select the desired printer on the left list and click <Add>. The selected printer will be added to your licensed printers list.

* <Remove>

To remove a printer from the licensed printers list, you simply need to select the desired printer on the right list and click <Remove>. The selected printer will be removed from your licensed printers list.

* <New Group>

Allow you to create a new printer group, see [Edit Printer Group](#) for more information on the printers group parameters and definitions.

* <Delete Group>

Allow you to delete an existing printer group. see [Edit Printer Group](#) for

more information on the printers group parameters and definitions.

* **<Edit Group>**

Allow you to edit an existing printer group. see [Edit Printer Group](#) for more information on the printers group parameters and definitions.

* **Allow printers by port**

When this checkbox is checked, the decision of whether a printer is licensed or not is made by port name in addition to the printer name. Therefore if you have a windows printer with a certain port, and you create another windows printer with the same port (from some implementation reasons), there is a need to add both printers to the licensed printers list (since both windows printers are actually the same physical printer – they have the same port), and still both printers will be licensed.

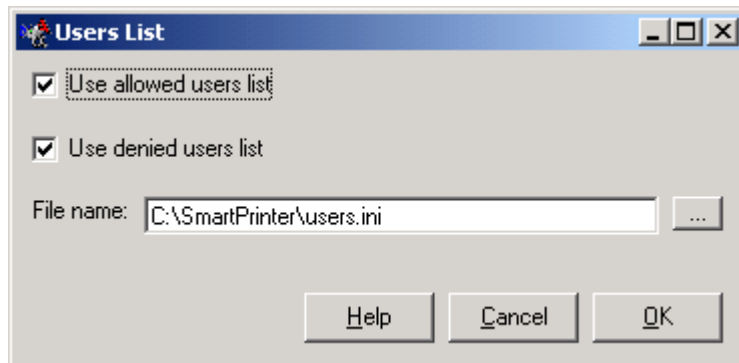


The PrintController will not allow you to add more printers to the licensed printers list from the maximum number of allowed printers.



The PrintController will not allow you to add the same printer twice to the licensed printers list.

PrintController Users Lists Dialog Box



The Users Lists dialog box enables you to block the printing via SmartPrinter to certain users and allow it to others. In order to do this, the SmartPrinter supports two users lists: Allowed Users and Denied Users. You may use any of the lists or even both lists at the same time.

*** Allowed Users List**

If you enable the use of the allowed users list, only users that appear on this list will be allowed to print via SmartPrinter. Any other user that will attempt to print will be blocked and the print job he sent will be cancelled.

*** Denied Users List**

If you enable the use of the denied users list, any user that appears on this list will be blocked and the print job he sent will be cancelled. All other users will be allowed.

Users List Format:

The format in which the users list is expected by the SmartPrinter is an INI file format, all the allowed users should appear under the section [AllowedUsers], and all the denied users should appear under the [DeniedUsers] section. The two sections must be in the same file. Each user should appear as the key and the value must be 1.

The separation to two lists is useful if you want to have a permanent allowed users lists (e.g. the list of all domain users) and a denied users lists that vary often (e.g. users that reached the end of the print quota)

An example of a users list file:

```
[AllowedUsers]
UserA=1
UserB=1
```

```
[DeniedUsers]
UserB=1
```

In this example only the user “UserA” will be allowed to print and all other users will be denied.



If both allowed and denied users lists are in use, the denied users list is stronger than the allowed users list, that is if a user appears on both lists it will be denied.

Different Users Lists for each printer:

It is also possible to make users lists that will be specific for just certain printers.

Under the section “PrinterToUsersListNum”, you can write printer names, and set a number for each printer. This number tells the SmartPrinter which number of users list (allowed and denied) to use for this printer. If the printer doesn’t appear in this section, the standard users list is in use.

An example of a multi users list file:

```
[PrinterToUsersListNum]
PrinterA=1
PrinterC=2
```

```
[AllowedUsers]
UserA=1
UserB=1
```

```
[DeniedUsers]
UserB=1
```

```
[AllowedUsers1]
UserA=1
UserB=1
```

```
[DeniedUsers1]
UserB=0
```

```
[AllowedUsers2]
UserA=1
UserB=1
```

```
[DeniedUsers2]
UserA=1
UserB=1
```

In this example for the printer “PrinterA” users list number 1 will be used, for the printer “PrinterC” users list number 2 will be used, and all other printer (e.g. “PrinterB”) will use the standard users list.

Edit Printer Group

In the edit printer group dialog box, you can select which printers will be in the group. And set all parameters concerning the group of printers. A group of printers is used when you want to split a job between several printers. The print job will be printed to the printers of the group in the order they appear on the list.

Edit printer group

Group name:

Minimum pages to divide:

☒ Search for fonts and copy it to all printers

☒ Search for the StartJob parameter and copy it to all printers

☒ Automatically identify StartJob parameter

Start of StartJob parameter:

1	2	3	4	5	6	7	8	9	10	11	12
Hex:											
Ascii:											

End of StartJob parameter:

1	2	3	4	5	6	7	8	9	10	11	12
Hex:											
Ascii:											

Available (and licensed) printers:

lumen
gest89
NewWhite
Fiery X12 Color Server PS

Printers in group:

lumen
gest89

Printer group parameters

* **Group Name**

The name of the group, it is available only if a new group is created, if you are editing an existing group, you can't change its name.

* **Minimum pages to divide**

This parameter allows you to instruct the SmartPrinter to divide a job only if it's bigger than a certain amount of pages, jobs smaller than this amount

will be printed to the first printer of the group.

*** Search for fonts and copy it to all printers**

This parameter decides whether the SmartPrinter will look for fonts in the print job and send them to all printers in the group

It is recommended you select this option.

*** Search for the StartJob parameter and copy it to all printers**

This parameter decides whether the SmartPrinter will look for StartJob parameter in the print job and send it to all printers in the group

The StartJob parameter will be searched according to the two following definitions. It is recommended you select this option.

*** Automatically identify StartJob parameter**

If you choose this check box, The SmartPrinter will search for a standard StartJob parameter in the print job, and share the same StartJob parameter between the different printers of the group. It is recommended you select this option.

*** Start of StartJob parameter**

If you choose to manually define the Start and End sequences of the StartJob parameter, Then the SmartPrinter will search for this ASCII sequence in the print job, and the StartJob parameter will be considered to start right after it.

If left empty, the SmartPrinter will treat the StartJob parameter as if it is starting at the beginning of the print job.

*** End of StartJob parameter**

If you choose to manually define the Start and End sequences of the StartJob parameter, Then the SmartPrinter will search for this ASCII sequence in the print job, and the StartJob parameter will be considered to end right after it. The bytes in this definition will be considered as a part of the StartJob parameter

*** Printers in the group**

Allow you to select which of the licensed printers will be a part of the group.



The PrintController will not allow you to add the same printer twice to the group.

PrintController Password Dialog Box

The change password dialog box is the place where you can set or edit an entrance password to the PrintController. If you wish to set a password, simply type it in the proper place, and then verify it. If you wish to remove the password, simply type in an empty password.

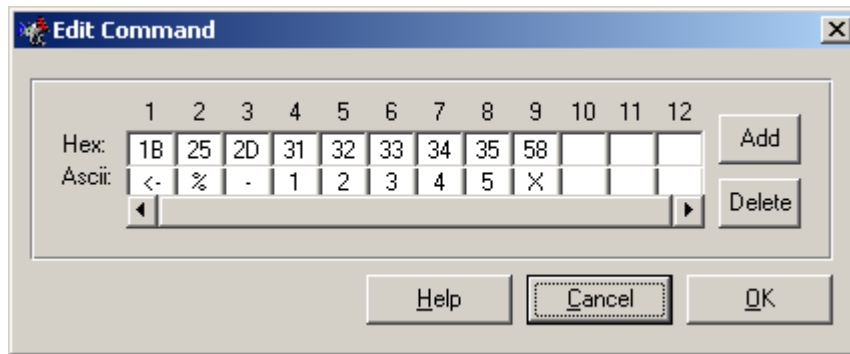
If a previous password was in order, and you wish to change it, you will be prompted to the old password before you could select the new password.



The password may only be composed of numbers.

Edit Command Dialog Box

The Edit Command dialog box is the place where you can edit some commands, such as the UEL(Universal Exit Language) or the Reset commands. There is sometimes a need to edit these commands since some printers demand a non-standard command.

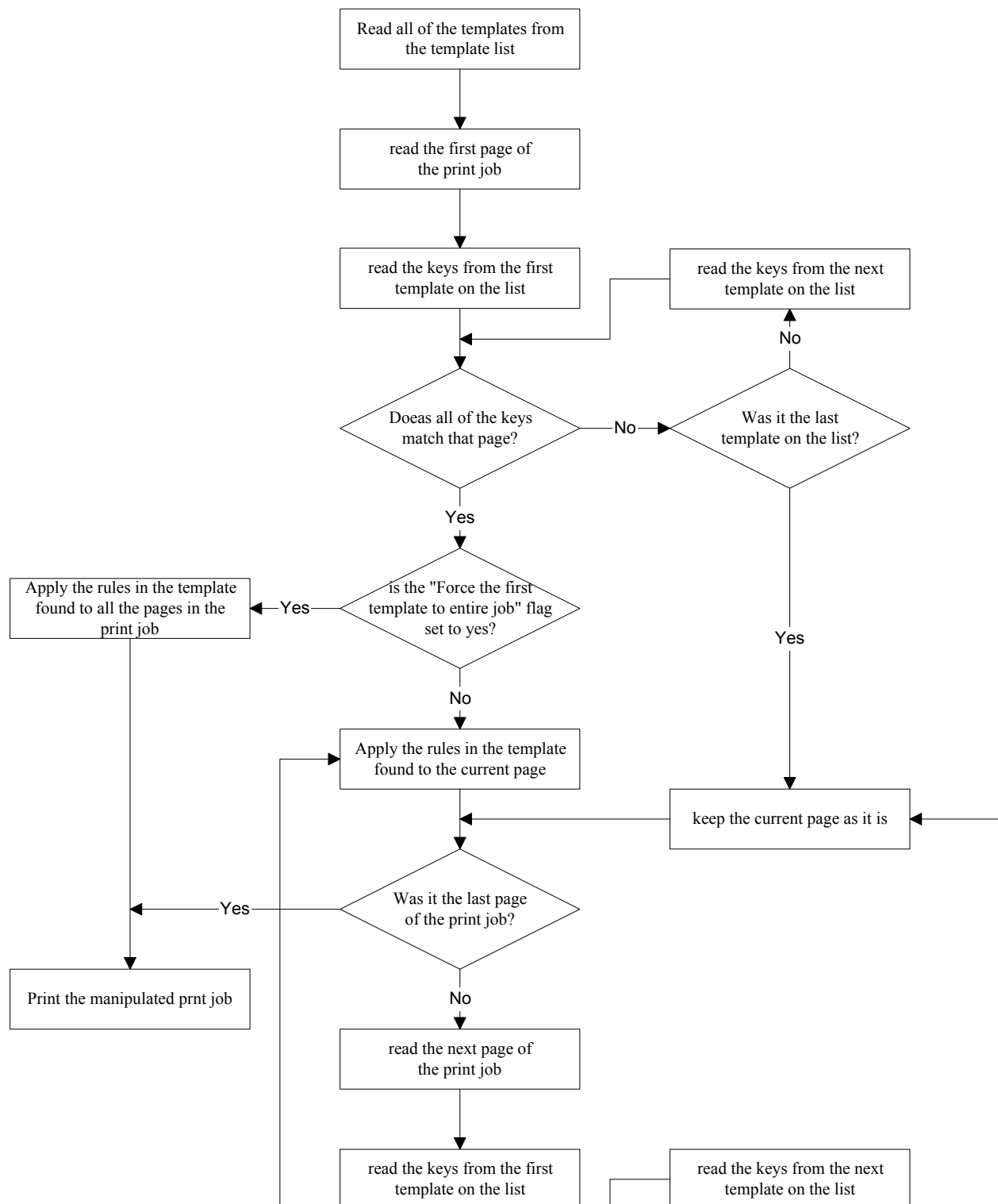


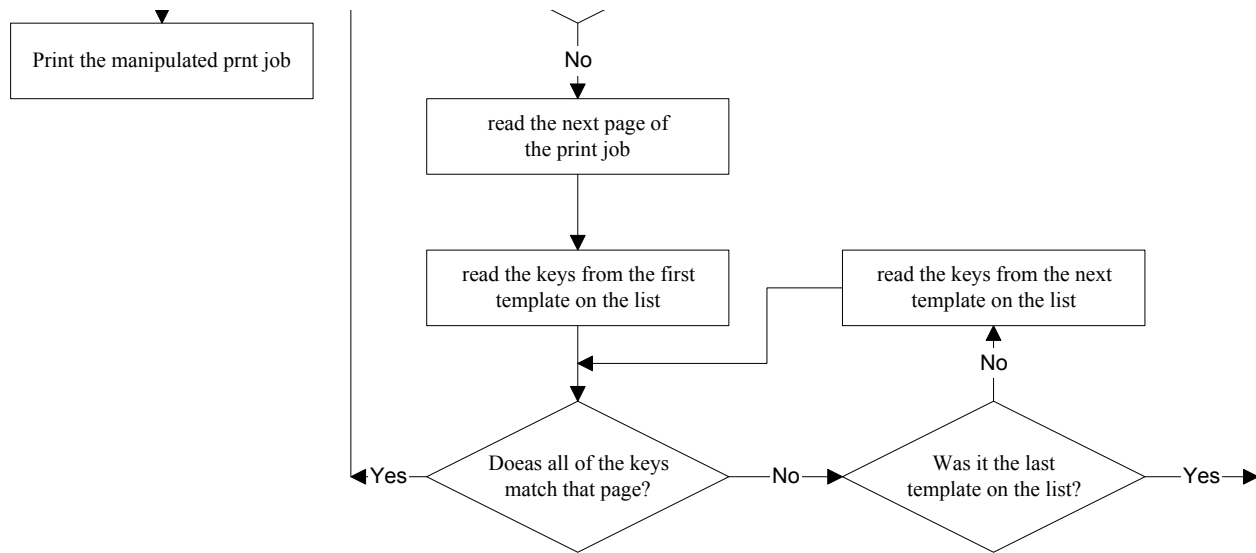
The editing of the sequence of characters the command is made of is done using the [Sequence Editor](#).

Template Identification Phase

In the template identification phase, the SmartPrinter processes the print job page by page and looks for a matching template for each page. The way that the SmartPrinter knows weather a template matches a page is according to the keys defined in the template. In order to achieve a match, all of the keys in the template must match exactly with the data on the page.

The process that occurs inside the template identification phase can be easily understood from the following scheme:





In order to edit a template, you must first [select a template](#), then edit it using the template editor interface.



In order to skip the template identification phase, check the Skip template identification phase checkbox in the [Options dialog box](#) in the PrintController.



The key identification phase is a “heavy” phase relatively to the general replacements phase, which is much “lighter” phase.

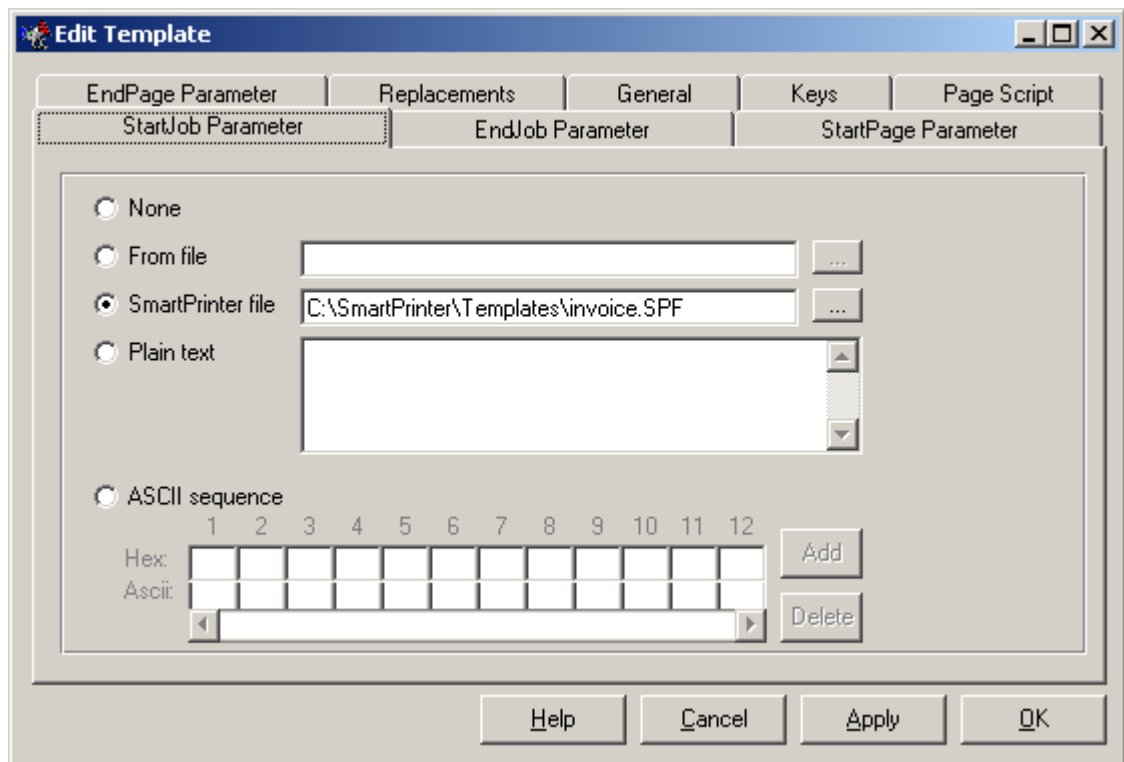
Edit Template - Description

A template is a set of rules you can determine, which will be activated only if the template is positively identified with a certain page of the print job. A page is identified with a template if all of the keys in the template match the data from the page in the print job.

The rules you can make in a template are:

- * Start Job
- * End Job
- * Start Page
- * End Page
- * Replacements
- * General
- * Keys
- * Page Script

Edit Template – Start Job



The Start Job parameter is a place where you can put information that will be added to the print job at its beginning. This information will be added only if this template was positively identified with the first page of the print job. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the beginning of the print job.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the beginning of the print job.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the beginning of the print job.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Edit Template – End Job

The screenshot shows the 'Edit Template' dialog box with the 'EndJob Parameter' tab selected. The dialog has a title bar with standard window controls. Below the title bar are tabs for 'EndPage Parameter', 'Replacements', 'General', 'Keys', and 'Page Script'. Under the 'General' tab, there are sub-tabs for 'StartJob Parameter', 'EndJob Parameter' (which is active), and 'StartPage Parameter'. The 'EndJob Parameter' section contains several options: 'None' (selected), 'From file' (with a text box and browse button), 'SmartPrinter file' (with a text box and browse button), and 'Plain text' (with a large text area). To the right of these options is a checkbox 'Run an external DLL file at the end of the job process.' with fields for 'DLL file name' and 'Procedure name inside DLL file'. Below these is an 'ASCII sequence' section with a grid of 12 columns and 2 rows (Hex and Ascii) and 'Add' and 'Delete' buttons. At the bottom are 'Help', 'Cancel', 'Apply', and 'OK' buttons.

The End Job parameter is a place where you can put information that will be added to the print job at its end. This information will be added only if this template was positively identified with the last page of the print job. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the end of the print job.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the end of the print job.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the end of the print job.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

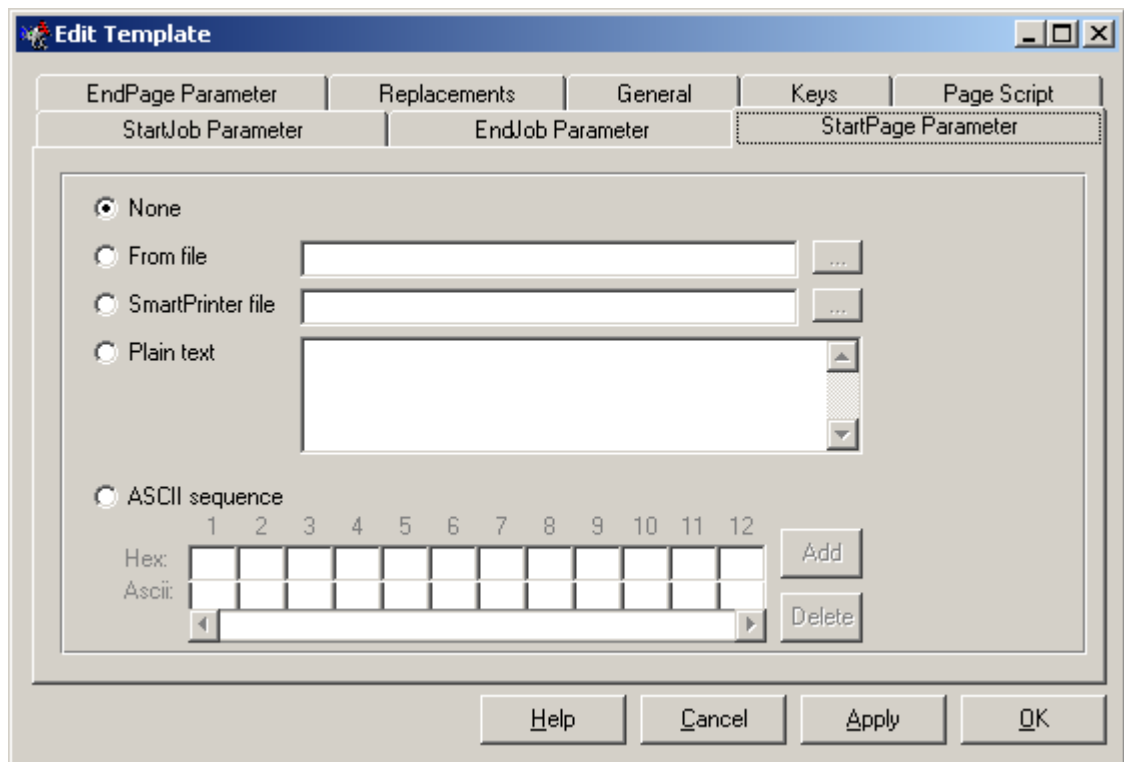
Run an external DLL file at the end of the job process:

When you check this checkbox, the procedure from the DLL file that was selected, will be called at the end of the job process. The procedure called must have one of the signatures that appear in the table below. The selection between the different procedure calls will be made according to the string you write at the “Procedure name” textbox. The different options are described in the table below:

Textbox content	Signature of the called procedure
ProcedureName	<code>void ProcedureName (void)</code>
ProcedureName,#GetJobNumber()	<code>void ProcedureName (int JobNumber)</code>

If the procedure called prints the print job to a printer, make sure to delete the job file from the spool directory, otherwise the print job may be printed twice.

Edit Template – Start Page



The Start Page parameter is a place where you can put information that will be added to the print job at the beginning of a certain page. This information will be added to the specific page only if this template was positively identified with that page of the print job. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the beginning of the page.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the beginning of the page.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the beginning of the page.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Edit Template – End Page

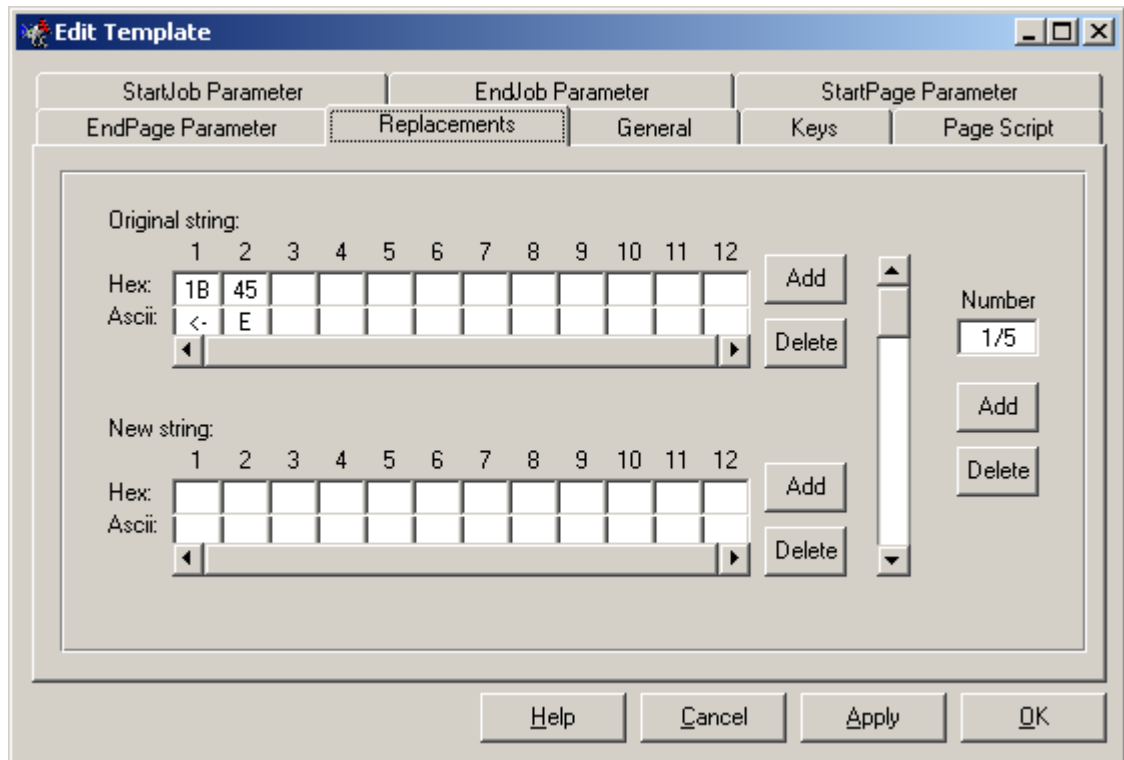
The screenshot shows the 'Edit Template' dialog box with the 'EndPage Parameter' tab selected. The dialog has a title bar with a standard Windows icon and window controls. Below the title bar are five tabs: 'StartJob Parameter', 'EndJob Parameter', 'StartPage Parameter', 'EndPage Parameter' (which is active and highlighted with a dotted border), 'Replacements', 'General', 'Keys', and 'Page Script'. The 'EndPage Parameter' tab contains several radio button options: 'None' (selected), 'From file', 'SmartPrinter file', and 'Plain text'. Each option has a corresponding text input field or a button to open a file. Below these is an 'ASCII sequence' section with a grid of 12 columns and 2 rows labeled 'Hex:' and 'Ascii:'. To the right of the grid are 'Add' and 'Delete' buttons. At the bottom of the dialog are four buttons: 'Help', 'Cancel', 'Apply', and 'OK'.

The End Page parameter is a place where you can put information that will be added to the print job at the end of a certain page. This information will be added to the specific page only if this template was positively identified with that page of the print job. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the end of the page.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the end of the page.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the end of the page.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Edit Template – Replacements

The replacements tab is a place where you can make replacement rules for the template. A replacement rule is an instruction to the SmartPrinter to replace a specific ASCII string with a different ASCII string.



Replacements parameters

- * **Original String**

The original string specifies the ASCII string that the SmartPrinter should look for.

Click [here](#) for more instructions on the way to edit the ASCII Sequence.

- * **New String**

The new string specifies the ASCII string that the SmartPrinter put instead of the original string.

Click [here](#) for more instructions on the way to edit the ASCII Sequence.

- * **Replacement Number**

The replacement number shows you in which replacement rule you are now. You can use the vertical scroll bar to navigate between the replacement rules for this template.

- * **<Add> button**

When you click this button, a new replacement rule is added to the template. The replacement rule is added before the replacement rule you were on. Use the [sequence editor](#) to edit the new replacement

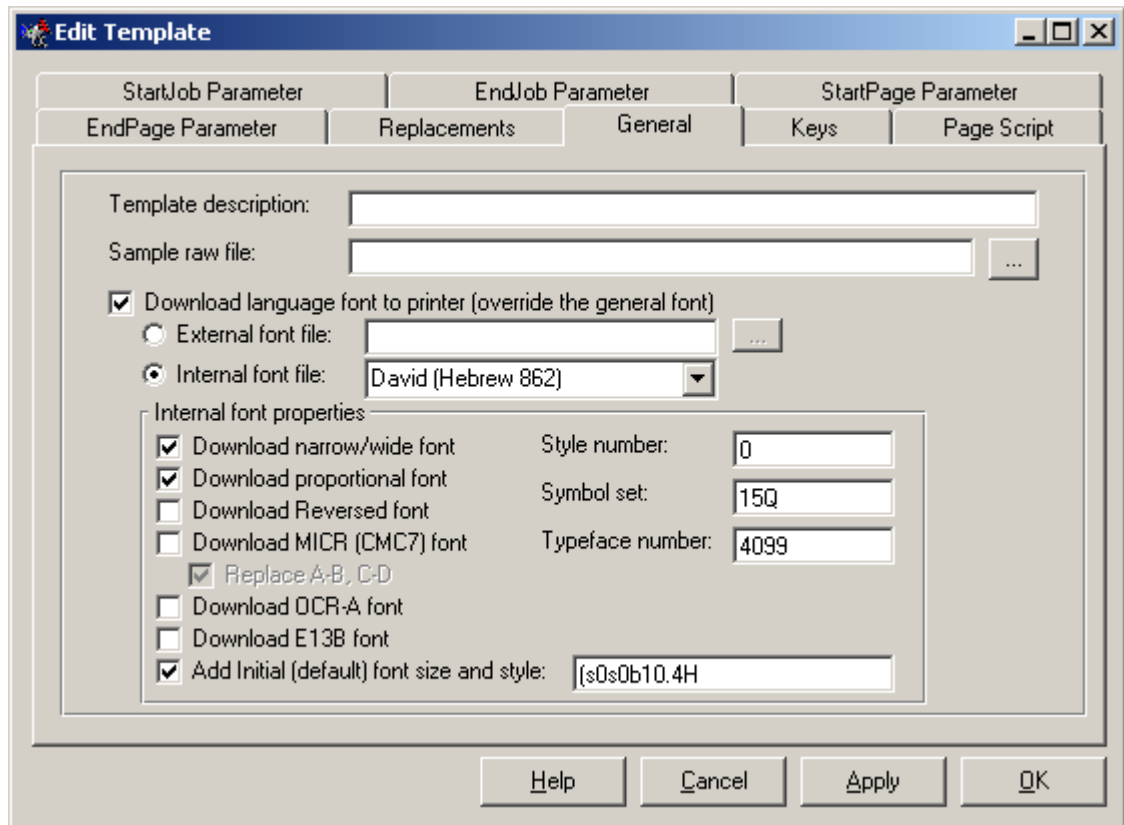
rule.

* **<Delete> button**

When you click this button, the replacement rule you were on is deleted from the template.

Edit Template – General

The general tab is a place where you can set some general parameters for the template.



General parameters

* Template Description

The template description is a text describes the template. This text appears next to the template file name in the [select template](#) dialog box, to assist you when you try to locate a specific template.

* Sample RAW File

In the sample raw file you can select a file containing a print job that is relevant to the template.

* Download language font to printer

When you check the download language font to printer checkbox, the font that will be downloaded to the printer when this template is positively identified by the first page of the print job will override the font defined in the [Font Management dialog box](#) in the PrintController. The properties of the font are exactly the same as explained in the [Font Management dialog box](#) in the PrintController.

Edit Template – Keys

The keys tab is a place where you can define the keys that will identify the template. A key is a combination of an ASCII string and a location for that string. A page positively identifies with a key if the data of the page contains the key's string in the exact key's location. A page positively identifies with a template if it positively identifies with all the keys in the template.

The screenshot shows the 'Edit Template' dialog box with the 'Keys' tab selected. The dialog has several tabs: StartJob Parameter, EndJob Parameter, StartPage Parameter, EndPage Parameter, Replacements, General, Keys, and Page Script. The 'Keys' tab is active, showing a checkbox for 'Make key valid when key is NOT found' (unchecked). Below this are input fields for 'Key Position': 'Top' (10), 'Left' (6), 'Down Offset' (0), and 'Right Offset' (0). The 'Key Text' section features a 12-column grid for Hex and Ascii values. The first six columns contain the hex values 49, 4E, 56, 4F, 49, 43 and their corresponding ASCII characters 'I', 'N', 'V', 'O', 'I', 'C'. The remaining six columns are empty. To the right of the grid are 'Add' and 'Delete' buttons. A vertical scrollbar is also present. At the bottom of the dialog are 'Help', 'Cancel', 'Apply', and 'OK' buttons.

	1	2	3	4	5	6	7	8	9	10	11	12
Hex:	49	4E	56	4F	49	43	45					
Ascii:	I	N	V	O	I	C	E					

Key parameters

- * **Key Not**
When this checkbox is checked, the key will be identified if the text of the key is NOT found in the specified location.
- * **Key Position**
The position of the key is the place in the page in which the SmartPrinter will look for the key text.
 - * **Top**
The line number (in rows) to look for the text.
 - * **Left**
The column number (in characters) to look for the text.
 - * **Delta Y**
The range of lines (in rows) to look for the text.

* **Delta X**

The range of columns (in characters) to look for the text.

Example: if the key has the following parameters: Top = 0, Left = 30, DeltaY = 2, DeltaX = 3 then the text will be searched in lines 0,1,2 and in the starting character of 30, 31, 32, 33. if you wish to search for the text only in a specific place then the DeltaY and DeltaX parameters should be set to 0 (this is the default).

* **Key Text**

The key text is the ASCII sequence that the SmartPrinter will look for at the location specified.

Click [here](#) for more instructions on the way to edit the ASCII Sequence.

* **Key Number**

The key number shows you in which key you are now. You can use the vertical scroll bar to navigate between the keys for this template.

If a template has more than one key, the template will be positively identified only if all of the keys match.

* **<Add> button**

When you click this button, a new key is added to the template. The key is added before the key you were on. Use the [sequence editor](#) to edit the new key.

* **<Delete> button**

When you click this button, the key you were on is deleted from the template.

Special key texts

* **TRUE Key**

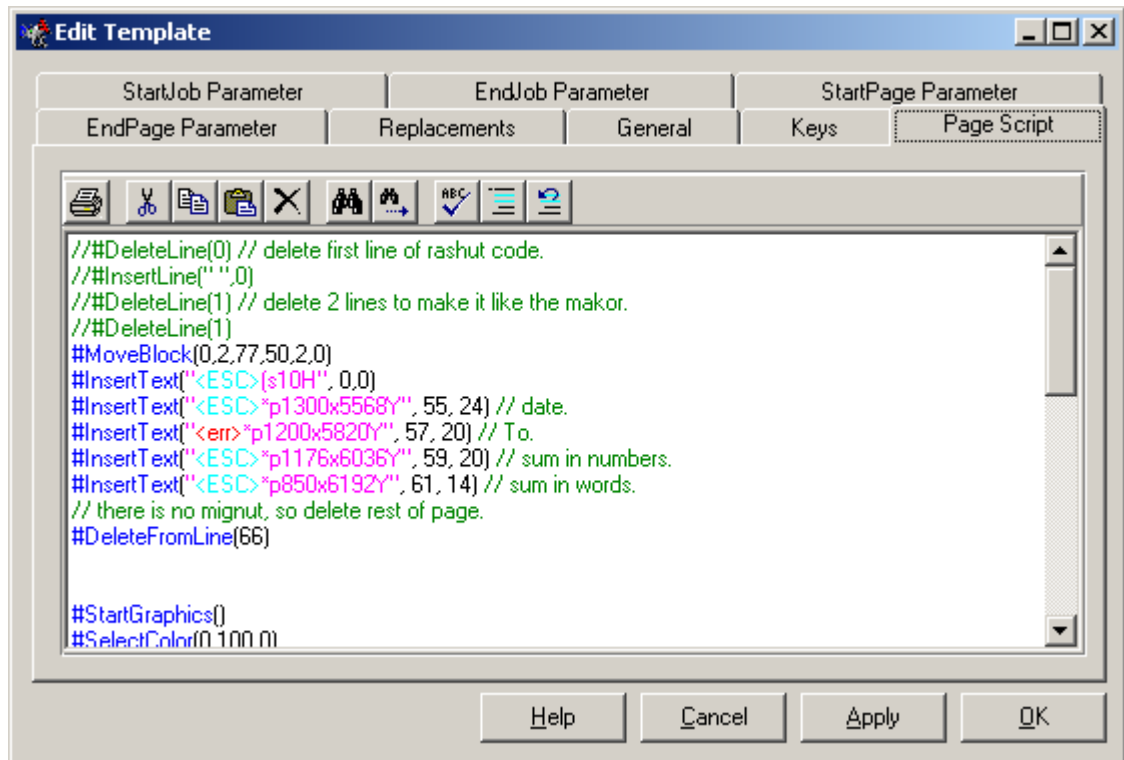
The TRUE key is a key that positively identifies with any page. In order to create a TRUE key, create a new key and write in the key text the string “##TRUE##”.

* **FALSE Key**

The FALSE key is a key that never positively identifies with a page. In order to create a FALSE key, create a new key and write in the key text the string “##FALSE##”.

Edit Template – Page Script

The Page Script tab is a place where you can write a script that will be preformed when a page that is positively identified with that template is printed.



The script editor has the following options:

- * **Print**
Allow you to print the script to any printer. If any part of the script is selected, only the selected part will be printed, otherwise all the script will be printed.
You may use the <CTRL> + <P> shortcut to activate this option.
- * **Cut/Copy/Paste/Delete**
Allow you to do standard clipboard operations
- * **Find**
Allow you to search the script for any phrase.
You may use the <CTRL> + <F> shortcut to activate this option.
- * **Find Next**
Searches the script for the last phrase that was already searched.
You may use the <F3> shortcut to activate this option.
- * **Spell Check**
Checks all the script for spelling errors regarding function names, etc...
Usually there is no need to do this, since the spell check is made while

typing.

You may use the <CTRL> + <S> shortcut to activate this option.

*** Comment**

Make all the selected lines commented (by adding “//” at the beginning of the line).

You may use the <CTRL> + <M> shortcut to activate this option.

*** UnComment**

Make all the selected lines uncommented (by removing the “//” from the beginning of the line).

You may use the <CTRL> + <U> shortcut to activate this option.

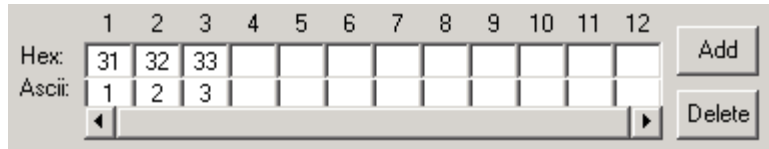
The following color code is in use:

- * Black = Normal text
- * Blue = Valid script functions
- * Green = comments
- * Cyan = Valid constants
- * Magenta = Strings
- * Red = Errors

A complete description of all the commands and constants in the Page Script can be found in the [Page Script Introduction](#).

Sequence Editor

The sequence editor is the tool in which you can edit ASCII strings. It is useful to create ASCII sequences containing unprintable characters such as CR (Hex=0D), LF (Hex=0A), Esc (Hex=1B) and so on.



Sequence editor parts:

- * **Hex row**

In the Hex row, you can see the hex values of the string being edited. When you enter a value to a cell in the Hex row, the appropriate ASCII character that matches the Hex value is printed in the cell below the Hex cell.

- * **ASCII row**

In the ASCII row, you can see the ASCII values of the string being edited. When you enter a value to a cell in the ASCII row, the appropriate Hex value that matches the ASCII character is printed in the cell above the ASCII cell.

- * **Numbering**

The numbering field shows what is your current location in the ASCII sequence being edited. You can use the horizontal scroll bar at the bottom of the sequence editor to navigate in the sequence edited.

- * **<Add> button**

When you click this button, a new character is added to the ASCII sequence. The character is added before the character you were on. The new character added has the Hex value of 0. You need to change that value to the desired character from the hex cell or from the ASCII cell.

- * **<Delete> button**

When you click this button, the character you were on is deleted from the ASCII sequence.

Page Script Introduction

Page script is a set of commands defined in the template editor, which will be operated on the page when the template is positively identified with that page:

The script commands are divided into several types:

- * [Arithmetic functions](#)
- * [String manipulation functions](#)
- * [Page Input/Output functions](#)
- * [Page Automatic analysis functions](#)
- * [Job Control functions](#)
- * [Counter functions](#)
- * [Variable functions](#)
- * [Flow Control functions](#)
- * [Graphic functions](#)
- * [Other functions](#)
- * [Constants](#)

To locate a specific function, please refer to the complete [Page Script functions index](#).

Each script function always start with the diez ‘#’ sign, then comes the function name, and then the function arguments inside brackets, when the arguments are separated from one another by the comma ‘,’ sign. This applies to all script functions besides the flow control functions – there the arguments are separated from one another by the semicolon ‘;’ sign.

A script function may be written as is in a single line, or may be divided between several line for easier understanding of the function. In that case, at the end of the line, the underscore ‘_’ sign must be the last sign of the line. Then the SmartPrinter will know to join these lines together to understand it.

You may write several script functions in the same line if you wish, when such a case occurs, these functions will simply be preformed one after the other. In several cases, you must do so, for example inside a flow control functions, if you wish to perform more then one function, write the functions one after the other, or in different lined but with the underscore sign at the end of each line.

Page Script functions index

A

- * [Abs](#)
- * [Add](#)
- * [AddMacro](#)
- * [And](#)
- * [Asc](#)
- * [Ascii2Ansi](#)
- * [Atoi](#)
- * [AutoCrBold](#)
- * [AutoFitFont](#)
- * [AutoTable](#)

C

- * [Chr](#)
- * [CloseFileEOF](#)
- * [ContainsHebrew](#)
- * [CopyBlock](#)
- * [CopyLine](#)
- * [CountDec](#)
- * [CountGet](#)
- * [CountInc](#)
- * [CountReSet](#)
- * [CountSet](#)

D

- * [Date](#)
- * [DecVar](#)
- * [DeleteTopEmptyLines](#)
- * [DeleteBottomEmptyLines](#)
- * [DeleteLeftEmptyColumns](#)
- * [DeleteRightEmptyColumns](#)
- * [DeleteBlock](#)
- * [DeleteColumns](#)
- * [DeleteEmptyLines](#)
- * [DeleteEmptyLinesFrom](#)
- * [DeleteFromLine](#)
- * [DeleteLine](#)
- * [DeleteText](#)
- * [Dim](#)
- * [Div](#)
- * [DoScriptCommand](#)
- * [DrawBarcode](#)
- * [DrawBox](#)
- * [DrawCircle](#)
- * [DrawCText](#)
- * [DrawFullBox](#)

* [DrawFullCircle](#)
* [DrawLine](#)
* [DrawRText](#)
* [DrawText](#)

E

* [Exit](#)
* [EndGraphics](#)
* [EngNumToWords](#)

F

* [FindTrigger](#)
* [FillChars](#)
* [FillSpaces](#)
* [For](#)

G

* [GetDaysFromBeginningOfYear](#)
* [GetDestinationPrinterName](#)
* [GetFromIniFile](#)
* [GetJobName](#)
* [GetJobNumber](#)
* [GetJobVar](#)
* [GetLine](#)
* [GetLineSize](#)
* [GetPageHeight](#)
* [GetPageMaxWidth](#)
* [GetPageNumber](#)
* [GetSourcePrinterName](#)
* [GetText](#)
* [GetUserName](#)
* [GetVar](#)
* [GetWordsUpTo](#)

H

* [Hex](#)

I

* [If](#)
* [IncVar](#)
* [InsertColumns](#)
* [InsertCommand](#)
* [InsertFile](#)
* [InsertLine](#)
* [InsertText](#)
* [InStr](#)
* [IsEmpty](#)
* [IsEqual](#)
* [IsFileEOF](#)
* [IsGreater](#)
* [IsSmaller](#)
* [Itoa](#)

L

* [Lcase](#)

	* <u>Left</u>
	* <u>Len</u>
	* <u>Logical2Visual</u>
	* <u>Ltrim</u>
M	
	* <u>MarkPageAsEndJob</u>
	* <u>MarkPageAsStartJob</u>
	* <u>Mid</u>
	* <u>Mod</u>
	* <u>MoveBlock</u>
	* <u>MsgBox</u>
	* <u>Mul</u>
N	
	* <u>Not</u>
	* <u>NumToWords</u>
O	
	* <u>OpenFileEOF</u>
	* <u>Or</u>
P	
	* <u>Power</u>
	* <u>PrintJobOnPrinter</u>
	* <u>PutText</u>
	* <u>PutTextInLines</u>
R	
	* <u>Random</u>
	* <u>ReadCharsFromFile</u>
	* <u>ReadLineFromFile</u>
	* <u>Replace</u>
	* <u>ReplaceInLines</u>
	* <u>ReplaceInOneLine</u>
	* <u>ReplaceInPage</u>
	* <u>ReverseStr</u>
	* <u>Right</u>
	* <u>RTrim</u>
	* <u>RunDll</u>
	* <u>RunExe</u>
S	
	* <u>SearchIniSection</u>
	* <u>SelectBackground</u>
	* <u>SelectColor</u>
	* <u>SelectFont</u>
	* <u>SetDestinationPrinterName</u>
	* <u>SetEmailServer</u>
	* <u>SetEmailFrom</u>
	* <u>SetEmailTo</u>
	* <u>SetEmailCC</u>
	* <u>SetEmailBCC</u>
	* <u>SetEmailSubject</u>
	* <u>SetEmailBody</u>

- * [SetEmailAdditionalAttachment](#)
- * [SetEmailAttachDocAsPDF](#)
- * [SetFaxBillCode](#)
- * [SetFaxDocName](#)
- * [SetFaxEmail](#)
- * [SetFaxName](#)
- * [SetFaxNumber](#)
- * [SetFaxSendComp](#)
- * [SetFaxSendDept](#)
- * [SetFaxSendName](#)
- * [SetJobFooter](#)
- * [SetJobHeader](#)
- * [SetJobName](#)
- * [SetJobVar](#)
- * [SetPageCopies](#)
- * [SetPageFooter](#)
- * [SetPageHeader](#)
- * [SetUserName](#)
- * [SetVar](#)
- * [SplitPageAfter](#)
- * [StartGraphics](#)
- * [StrCat](#)
- * [Sub](#)
- * [SubmitJobToSentinel](#)

T

- * [Time](#)
- * [Trim](#)

U

- * [Ucase](#)
- * [UnMarkPageAsEndJob](#)
- * [UnMarkPageAsStartJob](#)
- * [UseHpglText](#)

W

- * [While](#)
- * [WriteCharToFile](#)
- * [WriteLineToFile](#)
- * [WriteStringToFile](#)
- * [WriteToIniFile](#)

Arithmetic functions

Arithmetic function are function designed to make mathematical calculations. These function operate on integers and not on strings.

* **Abs**

Receives a number and returns it's absolute value.

Example:

`#Abs(-45) = 45`

`#Abs(45) = 45`

* **Add**

Adds the second argument to the first one and returns the result.

Example:

`#Add(4, 5) = 9`

* **And**

Makes a logical AND operation, return 1 only if receives 1 and 1, and 0 otherwise. If receives any other result but 0 or 1, it returns an empty value and an error is sent to the event log.

Example:

`#And(1,0) = 0`

`#And(1,1) = 1`

* **Div**

Divide the first argument in the second one and returns the result. If the second argument is 0 then the value an empty value is returned and an error is sent to the event log.

Example:

`#Div(20, 5) = 4`

* **Mod**

Divide the first argument in the second one and returns the reminder. If the second argument is 0 then the value an empty value is returned and an error is sent to the event log.

Example:

`#Mod(17, 5) = 2`

* **Mul**

Multiply the second argument with the first one and returns the result.

Example:

`#Mul(2, 3) = 6`

* **Not**

inverts the expression, return 1 if receives 0 and vice verse. If receives any other result but 0 or 1, it returns an empty value and an error is sent to the event log.

Example:

`#Not(1) = 0`

`#Not(0) = 1`

* **Power**

Raise the first argument by the power of the second one and returns the result.

Example:

`#Power(2, 3) = 8`

* **Or**

Makes a logical OR operation, return 0 only if receives 0 and 0, and 1 otherwise. If receives any other result but 0 or 1, it returns an empty value and an error is sent to the event log.

Example:

`#Or(1,0) = 1`

`#Or(0,0) = 0`

* **Sub**

Subtracts the second argument from the first one and returns the result.

Example:

`#Sub(2, 3) = -1`

`#Sub(3, 2) = 1`

String manipulation functions

String manipulation function are function designed to make manipulations on strings.

* **Asc**

Receives a string of one character and returns it's ASCII code. If a longer string is received, it returns an empty value and an error is sent to the event log.

Example:

`#Asc("A") = 65`

* **Chr**

Receives an ASCII code and returns the character which it represents. If a value that is not a valid ASCII value is received, it returns an empty value and an error is sent to the event log.

Example:

`#Chr(65) = "A"`

* **ContainsHebrew**

Returns the 1 if the string contains a Hebrew letter or 0 if it doesn't.

Example:

`#ReverseStr("ABC") = 0`

`#ReverseStr(#Hex("80")) = 1`

* **FillChars**

Fills the string in the first argument with a character from the second argument, from a certain direction (that is set by the third argument: 0=right align, 1=left align) to a certain length (that is set by the fourth argument).

Example:

`#FillChars("7", "0", 0, 5) = "00007"`

`#FillChars("7", "0", 1, 5) = "70000"`

* **FillSpaces**

Fills the string in the first argument with spaces, from a certain direction (that is set by the second argument: 0=right align, 1=left align) to a certain length (that is set by the third argument).

Example:

`#FillSpaces("7", 0, 5) = " 7"`

`#FillSpaces("7", 1, 5) = "7 "`

* **GetWordsUpTo**

Gets complete words from a string (the first argument) up to a certain location (that is set by the second argument)

Example:

`#GetWordsUpTo("This is a test for the function", 11) = "This is a "`

the 11th position of the string is the letter “e” in the word “test”, but since the function doesn’t cut words, it will return all the words up to the word “test” (not including)



This function is useful to cut a line into several parts without cutting words in the middle

* **Hex**

Converts a string of hex values into a regular string.

Example:

`#Hex(“48656C6C6F”) = “Hello”`

* **InStr**

Searches for the second string argument inside the first string argument and returns the position in which it was found, if it is not found then a 0 value is returned.

Example:

`#InStr(“abcde”, ”bc”) = 2`

* **Lcase**

Converts the string argument given into lowercase string.

Example:

`#LCASE(“AbCdEf”) = “abcdef”`

* **Left**

Returns the first x letters from the left of the first string argument, where x is the second argument.

Example:

`#Left(“ABCDE”, 3) = “ABC”`

* **Len**

Counts the length of the string argument given and return it.

Example:

`#Len(“abcde”) = 5`

* **Ltrim**

Trims the spaces from the left side of the string argument given.

Example:

`#LTrim(“ ABCDE ”) = “ABCDE ”`

* **Mid**

Returns the substring between letter number x to letter number y from the first string argument, where x is the second argument and y is the third argument.

Example:

`#Mid("abcdefgh", 3, 6) = "cdef"`

* **Replace**

Replace all occurrences of one substring (the second string argument) with another substring (the third string argument) at a specified string (the first string argument).

Example:

`#Replace("aaabbbcccbbaaa", "bbb", "XX") = "aaaXXcccXXaaa"`

* **ReverseStr**

Returns the reverse string of the original string.

Example:

`#ReverseStr("abc123") = "321cba"`

* **Right**

Returns the first x letters from the right of the first string argument, where x is the second argument.

Example:

`#Right("ABCDE", 3) = "CDE"`

* **RTrim**

Trims the spaces from the right side of the string argument given.

Example:

`#RTrim(" ABCDE ") = " ABCDE"`

* **StrCat**

Joins the two string arguments into one string and returns it.

Example:

`#StrCat("abcde", "bc") = "abcdebc"`

* **Trim**

Trims the spaces from both sides of the string argument given.

Example:

`#Trim(" ABCDE ") = "ABCDE"`

* **Ucase**

Converts the string argument given into uppercase string.

Example:

`#LCase("AbCdEf") = "ABCDEF"`

Page Automatic Analysis functions

Page Automatic Analysis functions are functions designed to make automatic input/output operations on the data page itself. All positions in the page are zero based, which means that they begin with 0 and not with 1. for example: the first line in the page is line 0, and the first character in a line is character 0.

* **AutoCrBold**

This function attempts to analyze the page and convert any repetition of lines on the same text to bolded text. In old line printers, there was an ability to actually print the same line twice using only the CR character at the end of the line. In the new laser printers this is impossible since the laser printing is accurate and the letter is printed exactly in the same place as the previous one – and no bold effect occurs. The function returns an empty value.

Example:

`#AutoCrBold()` => will convert repetitions of the same text as bold on all page.

`#AutoCrBold(8)` => will convert repetitions of the same text as bold on line 8 only.

`#AutoCrBold(8,25)` => will convert repetitions of the same text as bold on lines 8 through 25.

* **AutoFitFont**

This function attempts to fit the best font for the data to be printed. The parameter is the width of the paper used in the printer in cm. The Esc commands are being written at the top of the page (position: 0,0). The function returns an empty value.

Example:

`#AutoFitFont(21)` => will automatically fit the text to a portrait A4 paper (21 cm wide).

* **AutoTable**

This function attempts to analyze the page and convert any table like structure to a more visual table structure.

There are two versions of this function, the first has 2 string parameters, and the second has no parameters.

The parameters of the first version are 2 strings containing the characters by which the SmartPrinter will analyze the data (Vertical delimiters, Horizontal delimiters)

The second version simply uses the first version with the standard '+', '-', '|' characters, that means that `#AutoTable()` is equivalent to `#AutoTable("+|", "+-")`.

The function returns an empty value.

Example:

`#AutoTable("+|", "+-")` => will use the '+', '|' as vertical delimiters, and '+', '-' as horizontal delimiters, and will produce the following result:

No	Details	Value
1	ABC	10
2	DEF	15
3	aabbcc	142
4	xyz	63
5	xxxxxxxxxxxx	241
6	vv	94
Total		565



No	Details	Value
1	ABC	
2	DEF	
3	aabbccc	
4	xyz	
5	xxxxxxxxxxxx	
6	vv	
Total		565



Please note that since the '+' character is vertical delimiter and horizontal delimiter, it appears in both strings.



Please note that this operation is rather complicated and therefore relatively time expensive

Page Input/Output functions

Page input/output functions are functions designed to make input/output operations on the data page itself. All positions in the page are zero based, which means that they begin with 0 and not with 1. for example: the first line in the page is line 0, and the first character in a line is character 0.

* **AddMacro**

Adds the command which calls a macro, you need to load the macro file in the StartJob parameter, as a PCL macro file or as a [SmartPrinter file](#). The function arguments are (macro ID number, Top, Left). Top and left are the positions (in rows and columns) on the page to place the command. The function returns an empty value.

Example:

[#AddMacro](#)(1000, 0, 0) => add the command that will call macro with ID=1000 on the 0,0 position of the page.

* **CopyBlock**

Copies a block of text from one place on the page to the other. The function arguments are (Source Top, Source Left, Source Width, Source Height, Destination Top, Destination Left). The function returns an empty value.

Example:

[#CopyBlock](#)(0, 1, 6, 2, 3, 4) => copies the block that is on position: Top=0, Left=1, Width=6, Height=2 to position: Top=3, Left=4

* **CopyLine**

Copies a line of text from one row on the page to another. The function arguments are (Source row, Destination Row). The function returns an empty value.

Example:

[#CopyLine](#)(0, 10) => copies the line that is in row=0 to line that is in row=10

* **DeleteTopEmptyLines**

Deletes all the empty lines from the top of the page.

Example:

[#DeleteTopEmptyLines](#)()

* **DeleteBottomEmptyLines**

Deletes all the empty lines from the bottom of the page.

Example:

[#DeleteBottomEmptyLines](#)()

* **DeleteLeftEmptyColumns**

Deletes all the empty columns from the left of the page.

Example:

[#DeleteLeftEmptyColumns](#)()

* **DeleteRightEmptyColumns**

Deletes all the empty columns from the right of the page.

Example:

`#DeleteRightEmptyColumns()`

* **DeleteBlock**

Deletes characters of a specific block on the page. The lines that are manipulated by this command will be shifted to the left, but will not be shifted up. The function arguments are (Top, Bottom, Left, Right). The function returns an empty value.

Example:

`#DeleteBlock(2,5,10,20)` => deletes all characters from Left=10 to Left=20 in lines 2 to 5.

* **DeleteColumns**

Deletes characters of specific columns on the page. All lines will be shifted to the left, but there will not be a shift up. The function arguments are (Left, Right). The function returns an empty value.

Example:

`#DeleteColumns(10,20)` => deletes all characters from Left=10 to Left=20 in all the lines in the page.

* **DeleteEmptyLines**

Deletes a range of lines of text if they are empty (or contains only spaces) on a specific place on page. The function returns an empty value.

Example:

`#DeleteEmptyLines(1,4)` => deletes empty lines if exists on lines 1-4 of the page.

* **DeleteEmptyLinesFrom**

Deletes a range of lines of text if they are empty (or contains only spaces) from the specified line to the end of the page. The function returns an empty value.

Example:

`#DeleteEmptyLinesFrom(10)` => deletes empty lines if exists from line 10 to the end of the page.

* **DeleteFromLine**

Deletes all lines of text from a specific place on page to the end of the page. The function returns an empty value.

Example:

`#DeleteFromLine(5)` => deletes all lines from line 5 and on.

* **DeleteLine**

Deletes a line or a range of lines of text on a specific place on page. The function returns an empty value.

Example:

`#DeleteLine(1)` => deletes line 1 from the page

`#DeleteLine(1,4)` => deletes line 1-4 from the page



Please note that `#DeleteLine(1)` is identical to `#DeleteLine(1, 1)`.

* **DeleteText**

Deletes characters of specific line on the page. The line will be shifted to the. The function arguments are (Line, Left, Right). The function returns an empty value.

Example:

`#DeleteText(3, 10, 20)` => deletes all characters from Left=10 to Left=20 in line number 3.

* **GetLine**

Get a complete Line of text from the page.

Example:

`#GetLine(1)` = "Line 1 text" (Returns the text on line 1)

* **GetLineSize**

Return the number of characters in the line.

Example:

`#GetLineSize(10)` = x (Where x is the number of characters in line number 10)

* **GetPageHeight**

Return the number of lines in the page.

Example:

`#GetPageHeight()` = x (Where x is the number of lines in the page)

* **GetPageMaxWidth**

Return the length of the longest line in the page.

Example:

`#GetPageMaxWidth()` = x (Where x is the length of the longest line in the page)

* **GetPageNumber**

Return the number of the page in the job.

Example:

`#GetPageNumber()` = x (Where x is the number of the page in the job)

`#If(#IsEqual(#GetPageNumber(), 1);_`

`#InsertText("<ESC>&l5H", 0, 0);_`

`#InsertText("<ESC>&l3H", 0, 0);)`

=> Inserts a different page tray command for the first page and for all other pages.

* **GetText**

Get a text from a specific place on page and returns it as a string. The function arguments are (Top, Left, Width)

Example:

`#GetText(1,2,8)` = “Some Text...” (returns the text on position: Top=1, Left=2, Length=8)

* **InsertColumns**

Insert columns of spaces into the page. All lines will be shifted to the right, but there will not be a shift up. The function arguments are (Left, Right). The function returns an empty value.

Example:

`#InsertColumns(10,20)` => inserts 11 columns of blanks Left=10 to Left=20 in all the lines in the page.

* **InsertFile**

Insert a file on a specific place on page. The function returns an empty value.

Example:

`#InsertFile(“C:\SmartPrinter\Templates\Test.prn”, 10, 5)` => inserts the file “C:\SmartPrinter\Templates\Test.prn” to the page at Line 10, column 5.

* **InsertLine**

Insert a Line of text on a specific place on page. The function returns an empty value.

Example:

`#InsertLine(“abcd”, 1)` => inserts the text “abcd” as a new line on the page at position 1. The line that was at position 1 becomes position 2, and so on.

* **InsertText**

Insert a text on a specific place on a line. The function arguments are (Top, Left). The function returns an empty value.

Example:

`#InsertText(“abcd”, 1, 2)` => inserts the text “abcd” on position: Top=1, Left=2

* **MoveBlock**

Moves a block of text from one place on the page to the other. The function arguments are (Source Top, Source Left, Source Width, Source Height, Destination Top, Destination Left). The function returns an empty value.

Example:

`#MoveBlock(0, 1, 6, 2, 3, 4)` => moves the block that is on position: Top=0, Left=1, Width=6, Height=2 to position: Top=3, Left=4

* **PutText**

Put a text on a specific place on a line, and overwrite the existing characters of the line. The function arguments are (Top, Left). The function returns an empty value.

Example:

`#PutText(“abcd”, 1, 2)` => put the text “abcd” on position: Top=1, Left=2

* **PutTextInLines**

Put a text on a specific place on a range of lines, and overwrite the existing characters of the line. The function arguments are (Top, Bottom, Left). The function returns an empty value.

Example:

`#PutText("abcd", 12, 15, 2)` => put the text "abcd" on lines 12-15 position: Left=2

* **ReplaceInLines**

Replace all the occurrences of a string with another string at a specific range of lines. The function arguments are (Start Line Number, End Line Number, Original String, New String). The function returns an empty value.

Example:

`#ReplaceInLines(3, 6, "aaa", "bbb")` => will replace all the occurrences of the string "aaa" in lines 3-6 with the string "bbb"

* **ReplaceInOneLine**

Replace all the occurrences of a string with another string at a specific line. The function arguments are (Line Number, Original String, New String). The function returns an empty value.

Example:

`#ReplaceInOneLine(3, "aaa", "bbb")` => will replace all the occurrences of the string "aaa" in line 3 with the string "bbb"

* **ReplaceInPage**

Replace all the occurrences of a string with another string at the entire page. The function arguments are (Original String, New String). The function returns an empty value.

Example:

`#ReplaceInPage("aaa", "bbb")` => will replace all the occurrences of the string "aaa" in the page with the string "bbb"

* **SetPageCopies**

Set the number of copies, this page will be printed in. The function returns an empty value.

Example:

`#SetPageCopies(2)` => This page will be printed twice.

* **SetPageFooter**

Set the footer of the current page (EndPage Parameter) to be the second argument. While the first argument represents the number of the copy of the page to which the footer will be set.

Example:

`#SetPageFooter(1, "Testing")` ==> Set the EndPage parameter of the page to be the word "Testing" which means that this word will be added at the end of the page.

* **SetPageHeader**

Set the header of the current page (StartPage Parameter) to be the second argument. While the first argument represents the number of the copy of the page to which the header will be set.

Example:

```
#SetPageCopies(3)
#SetPageHeader(1, “(Esc)&l1H”)
#SetPageHeader(2, “(Esc)&l2H”)
#SetPageHeader(3, “(Esc)&l3H”)
```

This sequence of PageScript commands will cause the page to be printed if 3 copies, and for each copy to be printed from a different paper tray.

Graphic functions

Graphic functions are function designed to draw graphic elements on the paper. Whenever you want to draw graphics on the paper, you need to first call the [#StartGraphics](#) function, then call any graphic functions you like in the order you like the graphic elements to be painted, and at the end call [#EndGraphics](#) function which actually paints the elements. When using text functions such as [#DrawText](#), [#DrawRText](#), [#DrawCText](#) you must first select the font using [#SelectFont](#) command

* **DrawBarcode**

Draws a barcode. The function arguments are (Data, Top1, Left1, Top2, Left2, Code, Angle, Human Readable), as Data is the string of data to be coded, (Top1, Left1) are the coordinates of the top-left corner of the barcode and (Top2, Left2) are the coordinates of the bottom-right corner of the barcode, Angle is the angle in which to print the barcode (0, 90, 180 or 270), Human Readable determines weather or not to print the barcode's human readable text, if printed – it will be printed in the font selected by the [#SelectFont](#) command, and Code is a number setting the coding system in which to code the barcode. The function returns an empty value

Example:

[#DrawBarcode](#)("12345", 0, 1000, 300, 2000, 7, 90, 1) => draws a barcode from the point (0, 1000) to the point (300, 2000) containing the data "12345" coded in the "2 of 5" coding system, in 90 degree rotation angle, and with human readable text.

Codes:	<u>number</u>	<u>name</u>
	0	Code 128 Auto
	1	CODABAR
	2	Code 39
	3	Interleaved 2 of 5
	4	EAN8
	5	UPCE
	6	UPCA
	7	2 of 5
	8	Code 128 A
	9	Code 128 B
	10	Code 128 C
	11	EAN13
	12	Code 93



Not all coding systems will allow all data strings. Some systems only allow numbers while other allow numbers and alphanumeric, some allow only specific length of data while other allow variable length data.

* **DrawBox**

Draws a box. The function arguments are (X1, Y1, X2, Y2), as (X1, Y1) are the coordinates of the top-left corner of the box and (X2, Y2) are the coordinates of the bottom-right corner of the box. The function returns an empty value

Example:

[#DrawBox](#)(1000, 1500, 1500, 1700) => draws a box from the point (1000, 1500) to the point (1500, 1700).

* **DrawCircle**

Draws a circle. The function arguments are (X, Y, radius), as (X, Y) are the coordinates of center of the box and radius is the size of the circle's radius. The function returns an empty value

Example:

[#DrawCircle](#)(1000, 1500, 500) => draws a circle that its center is the point (1000, 1500) and its radius is 500.

* **DrawCText**

Draws a centered aligned text. The function arguments are ("String", X, Y, angle), as "string" is the string to be printed, (X, Y) are the coordinates that the string should be printed at, and angle if the angle (in degrees) that the text should be printed at. The function returns an empty value

Example:

[#DrawCText](#)("Testing", 1000, 1500, 45) => draws the text "Testing" at the point (1000, 1500) in a 45 degrees angle.

* **DrawFullBox**

Draws a Full box. The fill color is determined by the [#SelectBackground](#) function. The function arguments are (X1, Y1, X2, Y2), as (X1, Y1) are the coordinates of the top-left corner of the box and (X2, Y2) are the coordinates of the bottom-right corner of the box. The function returns an empty value

Example:

[#DrawFullBox](#)(1000, 1500, 1500, 1700) => draws a full box from the point (1000, 1500) to the point (1500, 1700).

* **DrawFullCircle**

Draws a full circle. The fill color is determined by the [#SelectBackground](#) function. The function arguments are (X, Y, radius), as (X, Y) are the coordinates of the center of the box, and radius is the size of the circle's radius. The function returns an empty value

Example:

[#DrawFullCircle](#)(1000, 1500, 500) => draws a circle that its center is the point (1000, 1500) and its radius is 500.

* **DrawLine**

Draws a line. The function arguments are (X1, Y1, X2, Y2), as (X1, Y1) are the coordinates of the first point of the line and (X2, Y2) are the coordinates of the second point of the line. The function returns an empty value

Example:

[#DrawLine](#)(1000, 1500, 1500, 1700) => draws a line from the point (1000, 1500) to the point (1500, 1700).

* **DrawRepeatedTextInBox**

Draws a repeated pattern of text inside a box. The function arguments are ("String", angle, distance, MaxTextBuffer, Top1, Left1, Top2, Left2), as (Top1, Left1) are the coordinates of the top-left corner of the box and (Top2, Left2) are the coordinates of the bottom-right corner of the box, "string" is the string to be printed, angle if the angle (in degrees) that the text should be printed at. distance is the distance between the lines of text, and MaxTextBuffer is the maximum number of characters to be used in the temporary buffer used by the function which should be large enough to contain the texts in the selected font on the desired side of the box. The function returns an empty value

Example:

`#DrawRepeatedTextInBox("Testing", 45, 100, 500, 1000, 1500, 1500, 1700)` => draws the text "Testing" at the box from the point (1000, 1500) to the point (1500, 1700) in a 45 degrees angle with a 100 distance between the lines.

* **DrawText**

Draws a text. The function arguments are ("String", X, Y, angle), as "string" is the string to be printed, (X, Y) are the coordinates that the string should be printed at, and angle if the angle (in degrees) that the text should be printed at. The function returns an empty value

Example:

`#DrawText("Testing", 1000, 1500, 45)` => draws the text "Testing" at the point (1000, 1500) in a 45 degrees angle.

* **EndGraphics**

Ends the current stream of graphic commands. The function arguments are (Top, Left), as Top is the position (in rows) to place the graphic stream code, and Left is the position (in characters) to place the graphic stream code. The function returns an empty value

Example:

`#EndGraphics(0, 0)` ==> collects all the graphic commands issued since the previous `#StartGraphics` command and generates the code to paint it, then inserts the code in the position selected (usually 0,0).

* **InsertCommand**

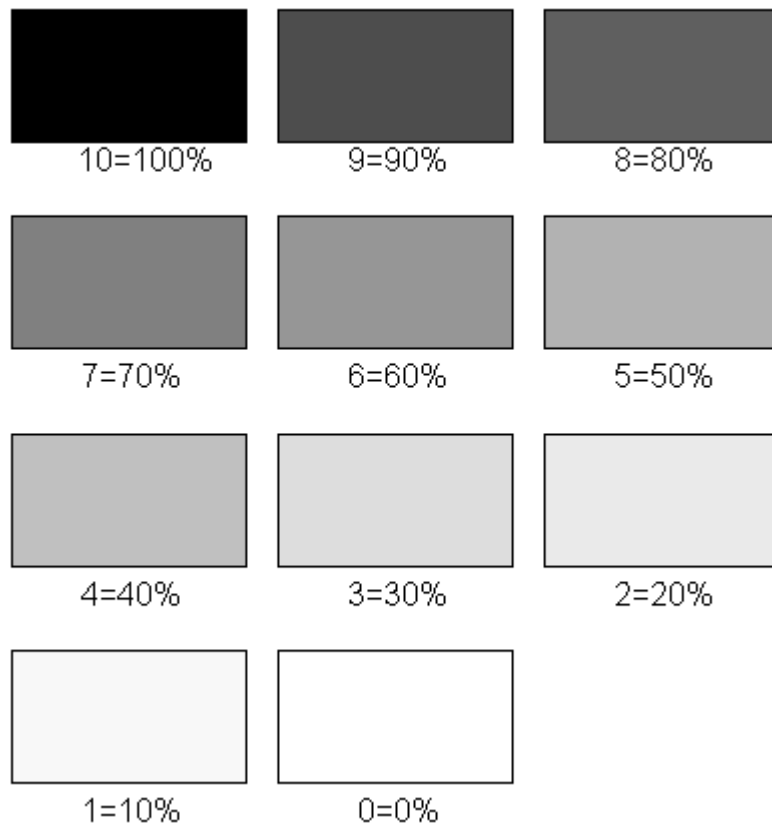
This function will inject a command (usually PCL command) into the stream of graphic commands created by the SmartPrinter.

Example:

`#InsertCommand("<ESC>(s10H")` ==> This will inject a the PCL command "<ESC>(s10H" into the steam of graphic commands.

* **SelectBackground**

Selects the fill background style that will be used in commands like `#DrawFullBox`, and `#DrawFullCircle`. The function argument is (shade) that will determine the level of shading according to the following table:



Example:

`#SelectBackground(3)`

`#DrawFullCircle(1000, 1500, 500)`

This sequence of PageScript commands will select 30% shade filling and draws a full circle that its center is the point (1000, 1500) and its radius is 500.



The range of the argument is 0-10 only. Any value lower than 0 will be considered as 0, and any value higher than 10 will be considered as 10.

* **SelectColor**

Selects the color that will be used in all drawing commands. The function arguments are (Red, Green, Blue) that will determine the color value according to standard RGB values

Example:

`#SelectColor(255, 0, 0)` => selects a red color.



The range of each argument is 0-255 only. Any value lower than 0 will be considered as 0, and any value higher than 255 will be considered as 255.



The `#SelectColor` command can be used with the `#SelectBackground` command to achieve special effects.



In most Black and White printers, any color but white will be interpreted as black.

* **SelectFont**

Selects the font that will be used in the [#DrawText](#) function. The function arguments are (CharSet, Spacing, Size, Weight, TypeFace), as CharSet is the character set of the font to be printed, Spacing indicates weather the font is fixed (=0) or proportional (=1) font, Size sets the size of the font to be printed (when the spacing is fixed, the size argument sets the CPI of the font, and when the spacing is proportional, the size argument sets the height of the font), Weight sets the stroke weight of the font: regular(=0), or bold (=3), TypeFace is the typeface of the font to be printed. The function returns an empty value

Example:

[#SelectFont](#)(497, 0, 6, 3, 4099) => selects the font that has a character set of 497 (15Q), fixed spacing, 6 CPI, bold stroke weight and typeface of 4099.



The font selected here must already exist at the printer (as a permanent font or downloadable font).



The formula to convert a character set to it's number (like 15Q to 497) is as follows: $(32 * \text{number}) + (\text{Asc}(\text{Char}) - 64)$. In our example: $(32 * 15) + (81 - 64) = 497$ (where the ASCII value of the letter Q = 81).

* **StartGraphics**

Starts a stream of graphic commands. The function has no arguments. Any graphic command written before this function will be ignored. The function returns an empty value

Example:

[#StartGraphics](#)()

* **UseHpglText**

Force the texts drawn by the [#DrawText](#), [#DrawCText](#), [#DrawRText](#) to be created using PCL/HPGL commands, the default is to use HPGL commands and in order to force PCL commands use 0 as the argument.

Example:

[#UseHpglText](#)(0).

Job Control functions

Job control functions are function designed to control the parameters related to the entire job.

* **GetDestinationPrinterName**

Returns the name of the destination printer the job will be printed on, as appears on the printer's list of the windows operating system.

Example:

`#GetDestinationPrinterName ()` => returns the name of the destination printer the job will be printed on.

* **GetSourcePrinterName**

Returns the name of the source printer the job was sent to, as appears on the printer's list of the windows operating system.

Example:

`#GeSourcePrinterName ()` => returns the name of the source printer was sent to.

* **GetJobName**

Returns the name of the current job.

Example:

`#GetJobName()` => Returns the name of the current job.

* **GetJobNumber**

Returns the unique number of the current job.

Example:

`#GetJobNumber()` => Returns the unique number of the current job.

* **GetUserName**

Returns the name of the user that printed the job, as received by the printer.

Example:

`#GetUserName()` => Returns the name of the user.

* **MarkPageAsEndJob**

Sets the current page to be an end of job, which means that after this page and its end page parameter, the end job parameter will be sent to the printer, then the start job parameters, and only then the next page with its start and end page parameters.

Example:

`#MarkPageAsEndJob()` => Sets the current page to be an end of job.



If this page has several copies (using the `#SetPageCopies` command) then the last copy of the page is considered as the end job indicator.

* **MarkPageAsStartJob**

Sets the current page to be a start of job, which means that before this page and its start page parameter, the end job parameter will be sent to the printer, then the start job parameters, and only then the page with its start and end page parameters.

Example:

`#MarkPageAsStartJob()` => Sets the current page to be a start of job.



If this page has several copies (using the `#SetPageCopies` command) then the first copy of the page is considered as the start job indicator.

* **PrintJobOnPrinter**

Prints the JOB, ORG or RPL files to another printer.

There are two versions of this function, the first has 1 string parameters, and the second has 3 string parameters. The first parameter is the name of the printer to which the job is to be printed, the second parameter can be either "ORG", "RPL" or "JOB" and it decides which file to print, and the third parameter is the new name of the printed job. The first version of the function simply prints the ORG file, and keeps the name of the job the same, that means that `#PrintJobOnPrinter("Printer1")` is equivalent to `#PrintJobOnPrinter("Printer1", "org", #GetJobName())`

Example 1:

```
#If(#IsEqual(#GetDestinationPrinterName(), "Printer1"));_  
#PrintJobOnPrinter("Printer2");;
```

=> If the destination printer was "printer1" prints the job to the printer "Printer2".

Example 2:

```
#If(#IsEqual(#GetJobName(), "PDF"));_  
#SetDestinationPrinterName("PDF");_  
#PrintJobOnPrinter(#GetSourcePrinterName(), "ORG", "PDF");;
```

=> If the name of the job is not PDF, then it will reprint the job to the same printer, but this time with the job name PDF, then in the second time the job will run through the system, it's name will be PDF, and then the destination printer will be PDF and therefore it will be printed to the PDF printer (assuming that the archive modul is licensed).



Never use this function without first checking to which printer the job was intended, and changing the printer, or check the job name and change it! Using this function carelessly can result in an endless loop of printing the same job over and over again!

* **SetDestinationPrinterName**

Sets the name of the destination printer the job will be printed on.

Example:

`#SetDestinationPrinterName("NewPrinter")` => Sets the name of the destination printer the job will be printed on to "NewPrinter".

* **SetEmailServer**

Set the email server for the current email. The email server must be a valid SMTP server that allows relay of emails from the server on which the SmartPrinter is installed. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailServer("mail.smartprinter.co.il")` ==> Set the email server of the email to be "mail.smartprinter.co.il".

* **SetEmailFrom**

Set the "From" field of current email. The "From" field must be an email address that will not be blocked by the email server and will be allowed to be relayed. This function only effect jobs that will eventually be sent via email instead of printed

Example:

`#SetEmailFrom("info@smartprinter.co.il")` ==> Set the email "From" field of the email to be "info@smartprinter.co.il".

* **SetEmailTo**

Set the "To" field of current email. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailTo("info@smartprinter.co.il")` ==> Set the email "To" field of the email to be "info@smartprinter.co.il".

* **SetEmailCC**

Set the "CC" field of current email. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailCC("info@smartprinter.co.il")` ==> Set the email "CC" field of the email to be "info@smartprinter.co.il".

* **SetEmailBCC**

Set the "BCC" field of current email. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailBCC("info@smartprinter.co.il")` ==> Set the email "BCC" field of the email to be "info@smartprinter.co.il".

* **SetEmailSubject**

Set the subject of current email. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailSubject("some subject")` ==> Set the email subject of the email to be "some subject".

* **SetEmailBody**

Set the body of current email. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailBody("some body")` ==> Set the email body of the email to be "some body".

* **SetEmailAdditionalAttachment**

If you want to attach more files to the email sent from the SmartPrinter server, you need to specify it here. If you want more then one file to be attached, you need to write all file names separated by a semicolon (;) character. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailAdditionalAttachment("c:\1.pdf; c:\2.pdf")` ==> Will cause the SmartPrinter to attach the files "c:\1.pdf" and "c:\2.pdf" to the outgoing email.

* **SetEmailAttachDocAsPDF**

If you want to cause SmartPrinter to attach the job file itself to the email as a PDF file, you need to use this function with a parameter of 1, if not then you need to use this function with a parameter of 0. If you don't use this function at all and the job is sent to an email, then the default value set by the PrintController interface will decide if the print job will be attached as a PDF. This function only effect jobs that will eventually be sent via email instead of printed.

Example:

`#SetEmailAdditionalAttachment(1)` ==> Will cause the SmartPrinter to attach the job file as a PDF file to the outgoing email.

* **SetFaxDocName**

Set the document name of the current fax job. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxDocName("SmartPrinter Document")` ==> Set the name of the fax recipient to be "SmartPrinter Document".

* **SetFaxEmail**

Set the email of the user that will receive a notification email if current fax will not pass right. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxEmail("kilany@smartprinter.co.il")` ==> Set the email address to send notifications to "kilany@smartprinter.co.il".

* **SetFaxName**

Set the name of the fax recipient of the current job. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxName("John Smith")` ==> Set the name of the fax recipient to be "John Smith".

* **SetFaxNumber**

Set the name of the fax recipient of the current job. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxNumber(#GetText(0, 0, 9))` ==> Set the number to which the fax job needs to be sent to the text in row 0 columns 0 to 8.



In order to send a print job via fax you need to change the destination printer to be "fax", you can do so by using the `#SetDestinationPrinterName("fax")` script command, or by using the [Printer Swapping dialog box](#).

* **SetFaxSendComp**

Set the company of the sender of the fax. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxSendComp("Smart Printing Solutions LTD.")` ==> Set the name of the fax sender to be "Smart Printing Solutions LTD.".

* **SetFaxSendDept**

Set the department of the sender of the fax. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxSendDept("Marketing")` ==> Set the name of the fax sender to be "Marketing".

* **SetFaxSendName**

Set the name of the sender of the fax. This function only effect jobs that will eventually be sent via fax instead of printed.

Example:

`#SetFaxSendName("John Smith")` ==> Set the name of the fax sender to be "John Smith".

* **SetJobFooter**

Set the footer of the current job (EndJob Parameter) to be the argument. This function is only meaningful if the current page being processed is the last page of the job, if it is not then there will be no effect.

Example:

`#SetJobFooter("<Esc>E")` ==> Set the EndJob parameter of the job to be the string "<Esc>E" which means that this string (which is a PCL reset command) will be added at the end of the job.

* **SetJobHeader**

Set the header of the current job (StartJob Parameter) to be the argument. This function is only meaningful if the current page being processed is the first page of the

job, if it is not then there will be no effect. If you want to control the job header from a page which is not the first page, you must uncheck the option "Print each page after finish processing it instead of entire job together" in the Tools Options dialog box.

Example:

[#SetJobHeader](#)("File:c:\Testing.prn") ==> Set the StartJob parameter of the job to be the file "c:\Testing.prn" which means that this file will be added at the beginning of the job.

* **SetJobName**

Sets the name of the current job.

Example:

[#SetJobName](#)("NewJobName") => Sets the name of the current job to be "NewJobName".

* **SetUserName**

Sets the name of the user that printed the job.

Example:

[#SetUserName](#)("NewUser") => Sets the name of the user to be "NewUser".

* **SubmitJobToSentinel**

This function instructs the SmartPrinter server to submit (or not) the print job into the Sentinel system after it finished processing the print job. This allows the two systems to work together.

Example:

[#SubmitJobToSentinel](#)(0) => Don't submit the print job to Sentinel.

[#SubmitJobToSentinel](#)(1) => Submit the print job to Sentinel.

[#SubmitJobToSentinel](#)(2) => Use the default value from the replace file.

* **UnMarkPageAsEndJob**

Unset the current page from being an end of job, please refer to [#MarkPageAsEndJob](#) function for more details.

Example:

[#UnMarkPageAsEndJob](#)() => Unset the current page from being an end of job.

* **UnMarkPageAsStartJob**

Unset the current page from being a start of job, please refer to [#MarkPageAsStartJob](#) function for more details.

Example:

[#UnMarkPageAsStartJob](#)() => Unset the current page from being a start of job.

Counter functions

Counters are special variables stored in an external memory, which means they keep its value between print job and between pages in the print job.

Counters can store only integer values.

* **CountDec**

Get the current value of the counter that is specified in the argument, return it, and then decrease it by 1.

Example:

`#CountDec("Count1") = 3` (assuming the value of the counter "Count1" was 3, then decreases its value to be 2).

* **CountGet**

Get the current value of the counter that is specified in the argument and return it.

Example:

`#CountGet("Count1") = 3` (assuming the value of the counter "Count1" was 3).

* **CountInc**

Get the current value of the counter that is specified in the argument, return it, and then increase it by 1.

Example:

`#CountInc("Count1") = 3` (assuming the value of the counter "Count1" was 3, then increases its value to be 4).

* **CountReset**

Set the current value of the counter that is specified in the argument and return it. Since it reset the counter, this function will always return 0.

Example:

`#CountReset("Count1") = 0` (reset the value of the counter "Count1").

* **CountSet**

Set the current value of the counter that is specified in the first argument to the value specified in the second argument, and return it.

Example:

`#CountSet("Count1", 5) = 5` (set the value of the counter "Count1" to be 5).

Variable functions

Variables are used to store information that is relevant to all the functions in the page. There are 2 types of variables: integer and string. The first thing you must do in order to use a variable is to define it, use the **#Dim** function to do so. Then you can set the variable's value using the **#SetVar** function, and get the variable's value using the **#GetVar** function.

When the page script is over, all of the variables are being erased.

* **Ascii2Ansi**

Convert a Hebrew ASCII string into a Hebrew Ansi string.

Example:

#Ascii2Ansi(#Hex("808182")) => will return the Hebrew Ansi string which is equivalent to #Hex("E0E1E2")

* **Atoi**

Convert a string into integer.

Example:

#Atoi("123") = 123

* **DecVar**

Decreases the value of the variable given as the argument by 1. If the variable contains a string value, the function will treat as if it contains the value 0 and decrease it by 1. The function returns an empty value.

Example:

#DecVar("x") ==> decreases the value of the variable "x" by 1.

* **Dim**

Defines a new variable and set its value to be 0.

Example:

#Dim("x") ==> defines the variable x and set it's value to be 0.

* **EngNumToWords**

Convert a numeric sum into words to use in checks. The function returns the sum in English in some currency. The function arguments are: (Numeric Sum, StarFill, NumLines, LineSize, MajorCoin, MinorCoin), Numeric sum is the number to be converted, StarFill can be either 0 or 1, and if 1 indicates to fill the spaces of the return string with stars, NumLines can be either 1 or 2, and means the number of lines to produce, LineSize means the length of each line, MajorCoin is a string with the major coin name (e.g. "Dollar"), MinorCoin is a string with the minor coin name (e.g. "Cent").

Example:

#EngNumToWords(123.45, 1, 2, 30, "Dollar", "Cent") => will return the English words description of 123.45 Dollars and Cents in two lines of 30 characters each, with stars fill on both sides of the text.

* **GetJobVar**

Get the value from a Job variable and returns it.

Example:

`#GetJobVar("x") = 3` (assuming the value of the job variable "x" was 3).

* **GetVar**

Get the value from a variable and returns it.

Example:

`#GetVar("x") = 3` (assuming the value of the variable "x" was 3).

* **IncVar**

Increases the value of the variable given as the argument by 1. If the variable contains a string value, the function will treat as if it contains the value 0 and increase it by 1. The function returns an empty value.

Example:

`#IncVar("x") ==>` increases the value of the variable "x" by 1.

* **Itoa**

Convert an integer into string.

Example:

`#Itoa(123) = "123"`

* **Logical2Visual**

Convert a Hebrew string from its logical display to its visual display.

Example:

`#Logical2Visual("אבג") => "גבא"`

* **NumToWords**

Convert a numeric sum into words to use in checks. The function returns the sum in Hebrew in NIS currency. The function arguments are: (Numeric Sum, StarFill, NumLines, LineSize), Numeric sum is the number to be converted, StarFill can be either 0 or 1, and if 1 indicates to fill the spaces of the return string with stars, NumLines can be either 1 or 2, and means the number of lines to produce, LineSize means the length of each line.

Example:

`#NumToWords(123.45, 1, 2, 30) =>` will return the Hebrew words description of 123.45 NIS in two lines of 30 characters each, with stars fill on both sides of the text.

* **SetJobVar**

Set the value of the job variable in the first argument to the value of the second argument. The function returns that value. The second argument may be a string or an integer. The function returns an empty value.

Example:

```
#If(#IsEqual(#GetPageNumber(), 1);#SetJobVar("Sum", 0);;)
#MsgBox(#GetJobVar("Sum"))
#SetJobVar("Sum", #Add(#GetJobVar("Sum"), #Atoi(#GetVar("TotalAmount"))));
#MsgBox(#GetJobVar("Sum"))
```

* **SetVar**

Set the value of the variable in the first argument to the value of the second argument.
The function returns that value. The second argument may be a string or an integer.
The function returns an empty value.

Example:

`#SetVar("x", 6) = 6` (set the value of the variable "x" to be 6).

`#SetVar("y", "Testing") = 6` (set the value of the variable "y" to be "Testing").



There is no need to define the variable before setting it, you can simply set a variable and if it doesn't exist yet, it will be defined automatically

Flow Control functions

Flow control function are used to perform operations that occur only in a certain condition. The operation to perform is a script function itself, and if you wish to perform several operations you need to simply write the function one after the other.

DoScriptCommand

The `#DoScriptCommand` function allows you to run script commands based on data from the print file

Example:

```
#DoScriptCommand(#GetLine(0));
```

Exit

The `#Exit` function allows you to exit the script for the current page. Any script command that comes after this function will not be preformed.

Example:

```
#Exit()
```

For

The `#For` function allows you to perform loops of script functions. The first part is an initialization command that is executed once. The second part is a condition that as long as it is true, the loop will continue. The third part is a termination command that is executed at the end of each loop. The fourth part is the body of the loop containing all the commands to be executed at each loop.

The syntax is as follows: `#For`(initialization command; condition; termination command; more script commands;)

Example:

```
#For(#SetVar("x", 0); #IsSmaller(#GetVar("x"),7); #IncVar("x");_  
#MsgBox(#GetVar("x"))_  
;)
```

The above example will display the 7 messages containing the numbers from 1 to 7.

*** If**

The `#If` function allows you to perform conditioned script functions. The syntax is as follows: `#If`(condition; true operation; false operation;). It means that the SmartPrinter will perform the script commands in the condition part, if the result returned by these functions is 1 and only if so, the functions in the true operation will be preformed, else the functions in the false operations will be preformed.

The syntax is as follows: `#If`(condition; true commands; false commands;)

Example:

```
#If(#IsEqual(1, #GetPageNumber());_  
#PutText("Page 1", 0, 0)_  
;_  
#PutText("Other Page", 0, 0)_  
;)
```

the above example will put the text "Page1" in position 0,0 of the page if the current page is

page 1 or the text “Other Page” in the same position if the current page is a different page.



Any of the true or false operation parts may be empty, it means that nothing will happen if that part is selected. If the condition part is empty, the false operation part will always be preformed.

RunDll

The **#RunDll** function allows you to run an external DLL from inside the SmartPrinter Script to perform various tasks.

The syntax is as follows: **#RunDll**(DllFileName, FunctionName, Paramater)

Example:

```
#RunDll("c:\test.dll", "SomeFunction", "Hello World")
```

RunExe

The **#RunExe** function allows you to run an external EXE software/utility from inside the SmartPrinter Script to perform various tasks.

The syntax is as follows: **#RunExe**(ExeFileName, Parameters, Visible, WaitForFinish)

Example:

```
#RunExe("c:\test.exe", "Some Data", 0, 0)
```

While

The **#While** function allows you to perform script functions as long as a certain condition is true.

The syntax is as follows: **#While** (condition; more script commands;)

Example:

```
#SetVar("x", 0)
#While(#Not(#IsEqual(#GetVar("x"), 5));_
#MsgBox("ABC")_
#SetVar("x", #Add(#GetVar("x"), 1))_
;)
```

The above example will display the Message “ABC” 5 times.



If you wish to perform several operations when a condition is true or false, you can simply write them one after the other in the appropriate part, as shown in the **#while** example.

Other functions

* **CloseFile**

Close a file that was opened using OpenFile function. The function returns an empty value.

Example:

`#CloseFile(#GetVar("hFile"))`

* **Date**

Returns the current date in the specified format. The applicable formats are: "dd/mm/yyyy", "dd.mm.yyyy", "mm/dd/yyyy", "mm.dd.yyyy".

Example:

`#Date("dd/mm/yyyy") = "01/01/2003"` (assuming that this is the current date).

* **FindTrigger**

Searches for a specific string in a certain column on each line of the page, and returns the place it found it. The function arguments are ("String", column, direction, number of instances, right offset). The first argument is the string to search for, the second argument is the column to search in, the third argument is the direction to look in (0 is from top to bottom of the page and 1 is from bottom to top), the forth argument is the number of appearance of the string and the fifth argument is the right offset in which the string may appear.

If the specified string is not found at all in the specified location, the value -1 is returned, and you can use the `#IF` command to verify it.

Example:

`#FindTrigger("AAA", 3, 0, 2, 3) ==>` will search for the text "AAA" on each line of the page from top to bottom and will return the 2nd line in which the text will appear of the 3rd, 4th or 5th columns.

* **GetDaysFromBeginningOfYear**

The `#GetDaysFromBeginningOfYear` function will return the number of days for either a specific date or the current date from the beginning of the year considering the different length of each month and considering a leap year.

The syntax is as follows: `#GetDaysFromBeginningOfYear(Year, Month, Day)`
or: `#GetDaysFromBeginningOfYear()` (for current date).

Example:

`#GetDaysFromBeginningOfYear(2014, 12, 31)` will return 365

`#GetDaysFromBeginningOfYear(2012, 12, 31)` will return 366

`#GetDaysFromBeginningOfYear()` will return number of days from beginning of this year.

* **GetFromIniFile**

Get a key value from an ini file (like the GetPrivateProfileString Windows API). The first argument is the ini file to read from, the second argument is the section name, the third argument is the key name and the fourth argument is the default value to return in case the action fails from any reason. The return value is the required value or the default specified value

Example:

`#GetFromIniFile("c:\temp\temp.ini", "general", "counter", "0") ==>` will read the key

“counter” from the section “general” in the file “c:\temp\temp.ini” and return its value, if the key wasn’t found - the value “0” will be returned

* **IsEmpty**

Checks whether the string argument is empty or not and return 1 if it is empty and 0 otherwise. This function will also return 1 when the string contains only space characters.

Example:

#IsEmpty(“123”) = 0

#IsEmpty(“”) = 1

#IsEmpty(“ ”) = 1

* **IsEqual**

Compares two expressions and return 1 if they are equal and 0 otherwise. This function can receive either two integer arguments or two string arguments.

Example:

#IsEqual(1,3) = 0

#IsEqual(1,1) = 1

* **IsFileEOF**

Will return 0 if the file is at its end position yet or 1 if the file has ended.

Example:

#IsFileEof(#GetVar (“hFile”)) ==> will return 0 if the file is at its end position yet or 1 if the file has ended.

* **IsGreater**

Compares two expressions and return 1 if the first is greater then the second and 0 otherwise. This function receives two integer arguments.

Example:

#IsGreater(1,3) = 0

* **IsSmaller**

Compares two expressions and return 1 if the first is smaller then the second and 0 otherwise. This function receives two integer arguments.

Example:

#IsSmaller(1,3) = 1

* **MsgBox**

Show a Message Box with the message written. The function’s argument may be an integer or a string. The function returns an empty value.

Example:

#MsgBox(“Hello”) ==> shows a message with the content “Hello”.

#MsgBox(7) ==> shows a message with the content 7.

* **OpenFile**

Open a file for reading or writing and returns a handle to this file. The handle is used in

order to read or write data to the file. The handle must be used to close the file at the end of the process. The first argument is the name of the file to open, the second argument is "R" if you want to open the file for reading, "W" if you want to open the file for writing or "A" if you want to open the file for append writing. The function returns a handle to the open file or 0 if failed. If the file is opened for writing and the file already exists, it will be replaced, if it doesn't exist, it will be created.

Example:

`#OpenFile("c:\temp\temp.txt", "R")` ==> will open the file "c:\temp\temp.txt" for reading.

* **Random**

Returns a random number which its value is between the value of the first argument and the second argument (including these values).

Example:

`#Random(3,7)` = 4 (Returns a random number which can be any of the following numbers 3,4,5,6,7).

* **ReadCharsFromFile**

Read a certain amount of characters from an opened file. The first argument is the handle to the file to read from, the second argument is the number of characters needed to read from this file. The function returns the read data.

Example:

`#ReadCharsFromFile(#GetVar ("hFile"))` ==> will read 5 characters from the current position of the opened file and return it.

* **ReadLineFromFile**

Read an entire line from an opened file. The first argument is the handle to the file to read from. The function returns the read data.

Example:

`#ReadLineFromFile(#GetVar ("hFile"))` ==> will read until the end of the line from the current position of the opened file and return it.

* **SearchIniSection**

Searches for a value in a section of an ini file. The first argument is the ini file to read from, the second argument is the section name, the third argument is the value to look for. The function reads all numbered keys (1, 2, 3 ...) from that section and returns the index in which the value was found. If the value was not found then 0 is returned.

Example:

`#SearchIniSection("c:\temp\temp.ini", "general", "abcde")` ==> will read keys 1, 2, 3... from the section "general" in the file "c:\temp\temp.ini" and return the key number (index) that its value was "abcde", if the key wasn't found - the value 0 will be returned.

* **SplitPageAfter**

Splits a page that is longer than a certain amount of lines to sections that are separated by FormFeed character, additionally an init string is added at the start of each such section. The first argument is the string to add at the start of each section, the second argument is the maximum number of lines per section. The function returns an empty value.

Example:

`#SplitPageAfter("aaa", 64) ==>` will split any page longer than 64 lines to sections that are 64 lines or less, and add FormFeed character between the sections. Additionally the string "aaa" will be added in the beginning of each section.

*** Time**

Returns the current time in the specified format. The applicable formats are: "hh:mm:ss", "hh:mm".

Example:

`#Time("hh:mm:ss") = "12:00:00"` (assuming that this is the current time).

*** WriteCharToFile**

Write a specific character to an opened file, according to the ASCII value of that character. This function is useful to write unprintable characters to the file. The first argument is the data to write as a decimal value of the character, and the second argument is the handle to the file to write to. The function returns the number of characters written.

Example:

`#WriteCharToFile(27, #GetVar ("hFile")) ==>` will write the line "ABC" to the opened file.

*** WriteLineToFile**

Write an entire line to an opened file, including the carriage return and line feed characters at the end of the line. The first argument is the data to write, and the second argument is the handle to the file to write to. The function returns the number of characters written.

Example:

`#WriteLineToFile("ABC", #GetVar ("hFile")) ==>` will write the line "ABC" to the opened file.

*** WriteStringToFile**

Write a string to an opened file. The first argument is the data to write, and the second argument is the handle to the file to write to. The function returns the number of characters written.

Example:

`#WriteStringToFile("ABC", #GetVar ("hFile")) ==>` will write the line "ABC" to the opened file.

*** WriteToIniFile**

Write a value to an ini file (like the WritePrivateProfileString Windows API). The first argument is the ini file to write to, the second argument is the section name, the third argument is the key name and the fourth argument is the value to write. The function returns an empty value.

Example:

`#WriteToIniFile("c:\temp\temp.ini", "general", "counter", "3") ==>` will write the value "3" to the key "counter" from the section "general" in the file "c:\temp\temp.ini"

Constants

Constants are expressions that can be placed anywhere in the script, when the SmartPrinter will do the script, it will replace these constants with the actual data.

Example for using the constants:

`#InsertText("<ESC>&l00", 0, 0) ==>` will insert the Escape character (0x1B) instead of the <ESC> constant and then continue the script as usual .

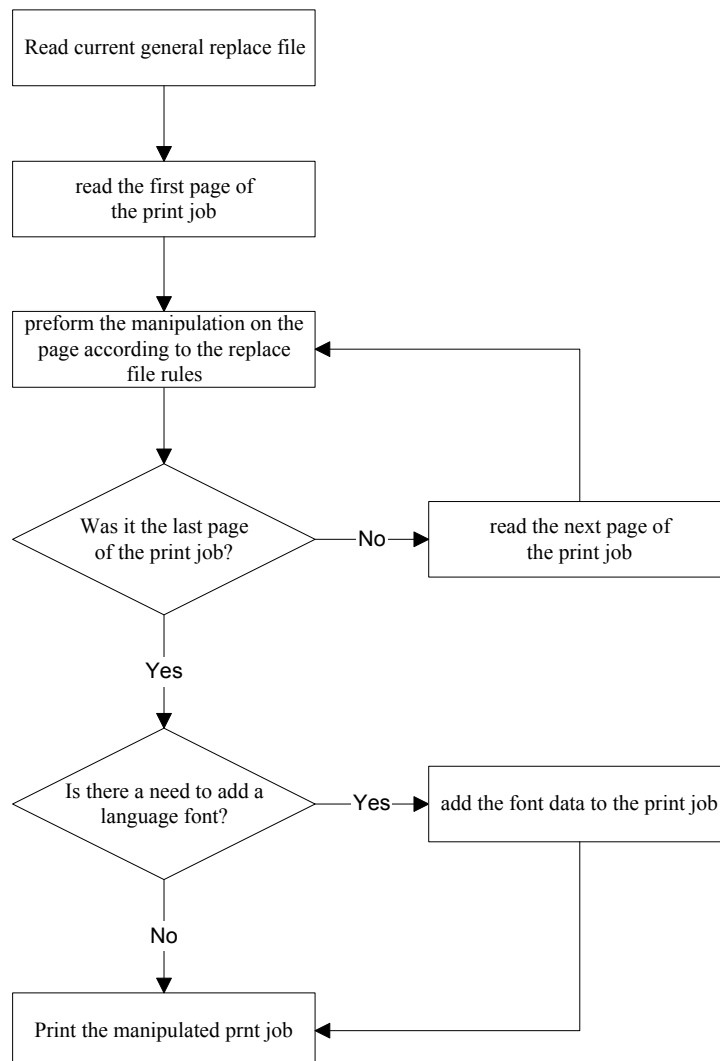
The following constants are available:

Constant	Real Value
<ESC>	The Escape character (0x1B)
<PAR>	The parenthesis character (“ ”)
<FF>	The Form Feed character (0x0C)
<CR>	The Carriage Return character (0x0D)
<LF>	The Line Feed character (0x0A)

General Replacements Phase

In the general replacements phase, the SmartPrinter processes the print job page by page and activates the rules defined on the replace file on the print job.

The process that occurs inside the general replacements phase can be easily understood from the following scheme:



In order to edit a the replace file, you need to use the [Manage Replace Files dialog box](#) in the PrintController, then select the desired replace file from the list, click the <Edit> button and edit it using the replace file editor interface.



In order to skip the general replacements phase, check the Skip general replacements phase checkbox under the [Options dialog box](#) in the PrintController.



The general replacements phase is a “light” phase relatively to the key



identification phase, which is a much "heavier" phase.

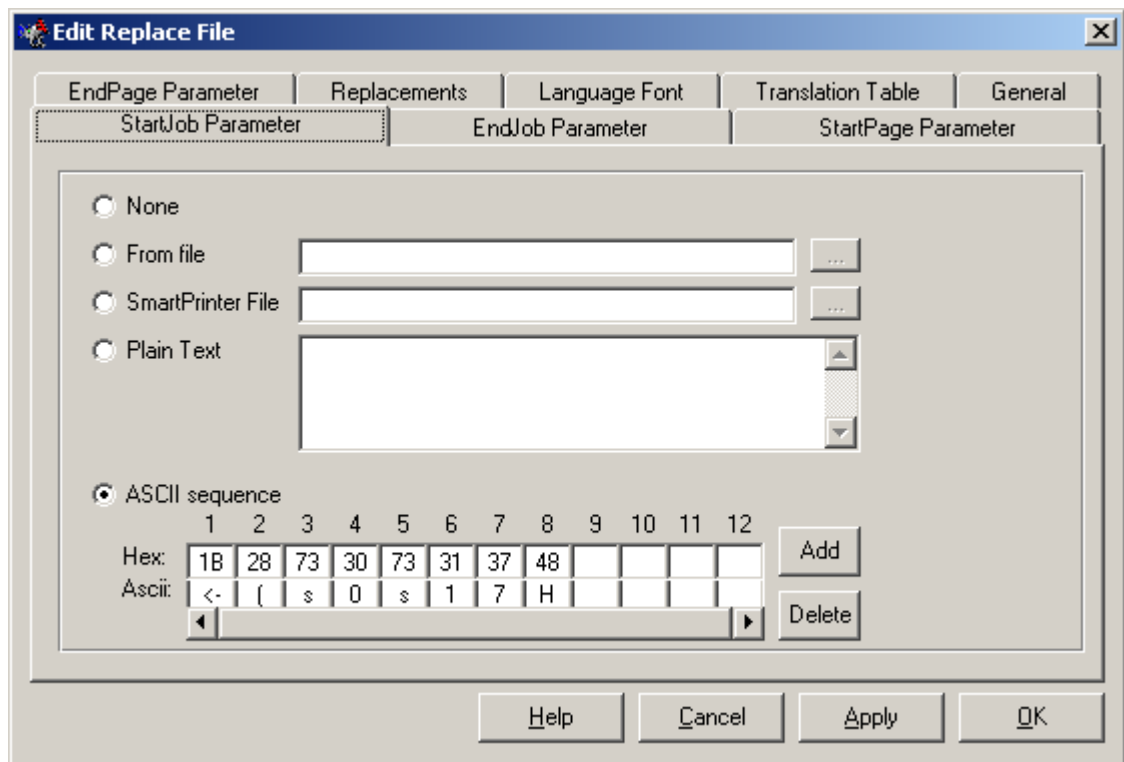
Edit Replace File - Description

A replace file is a set of rules you can determine, which will be activated on each page of the print job.

The rules you can make in a the replace file are:

- * **Start Job**
- * **End Job**
- * **Start Page**
- * **End Page**
- * **Replacements**
- * **Language Font**
- * **General**

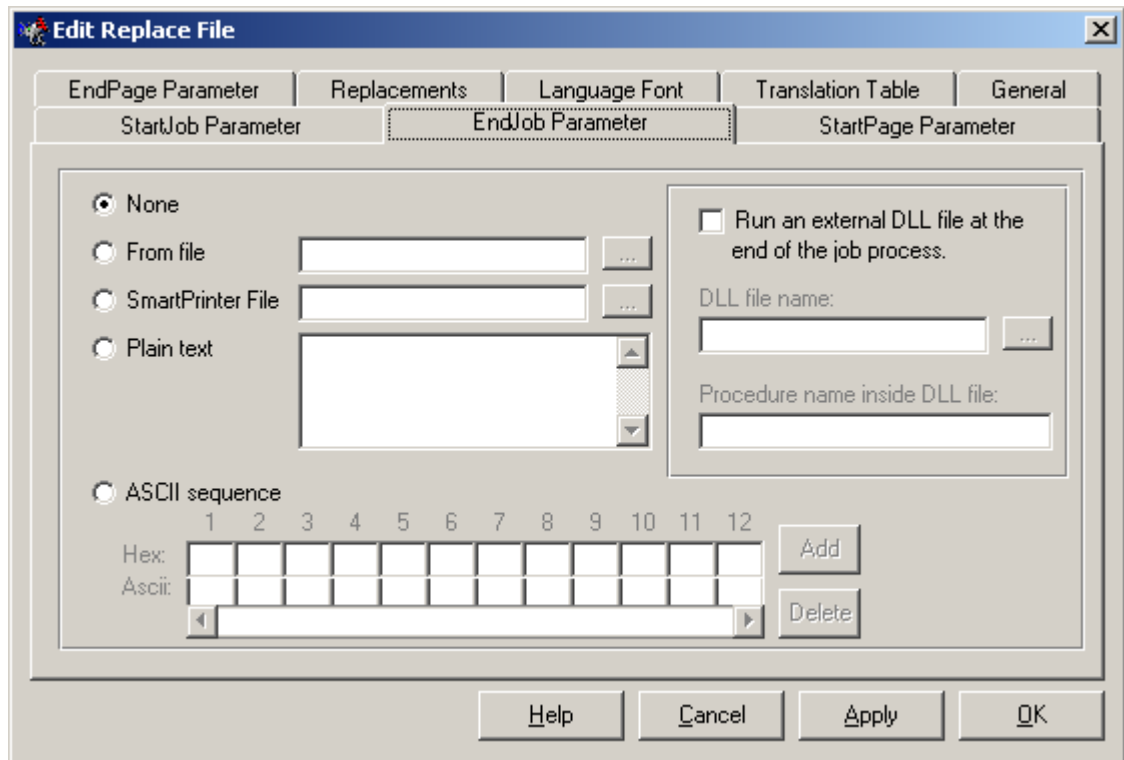
Edit Replace File – Start Job



The Start Job parameter is a place where you can put information that will be added to the print job at its beginning. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the beginning of the print job.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the beginning of the print job.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the beginning of the print job.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Edit Replace File – End Job



The End Job parameter is a place where you can put information that will be added to the print job at its end. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the end of the print job.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the end of the print job.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the end of the print job.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Run an external DLL file at the end of the job process:

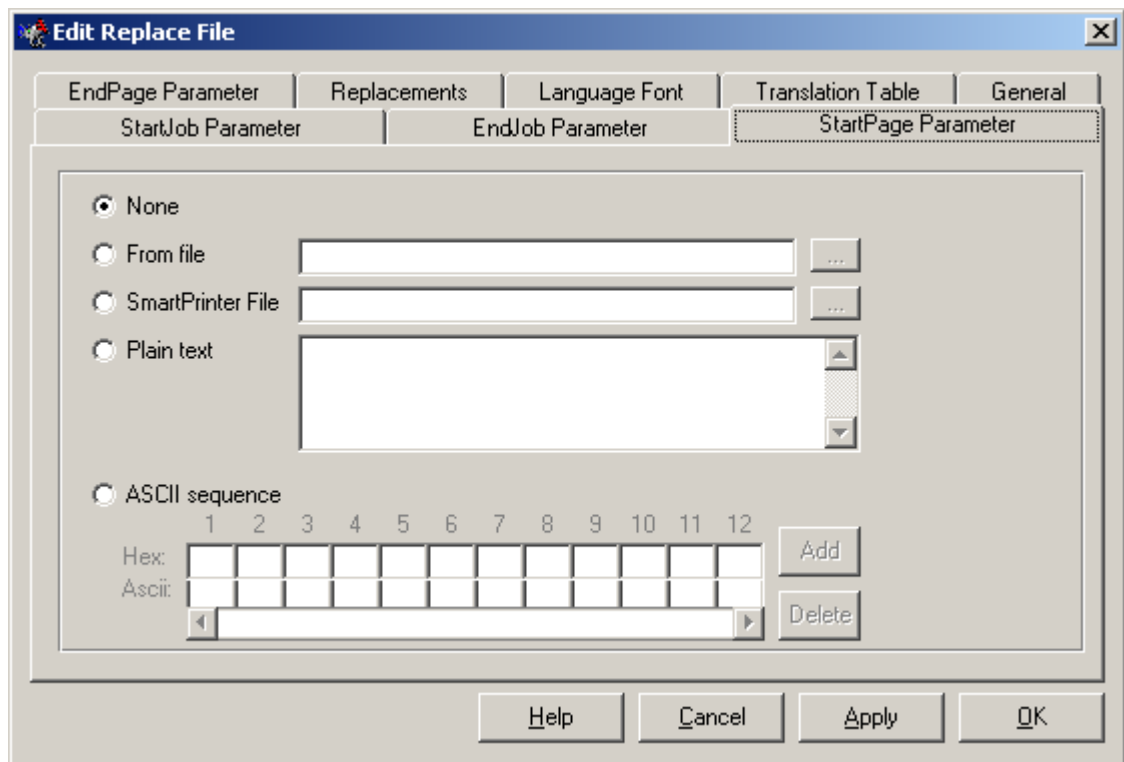
When you check this checkbox, the procedure from the DLL file that was selected, will

be called at the end of the job process. The procedure called must have one of the signatures that appear in the table below. The selection between the different procedure calls will be made according to the string you write at the “Procedure name” textbox. The different options are described in the table below:

Textbox content	Signature of the called procedure
ProcedureName	<code>void ProcedureName (void)</code>
ProcedureName,#GetJobNumber()	<code>void ProcedureName (int JobNumber)</code>

If the procedure called prints the print job to a printer, make sure to delete the job file from the spool directory, otherwise the print job may be printed twice.

Edit Replace File – Start Page



The Start Page parameter is a place where you can put information that will be added to the print job at the beginning of each page. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the beginning of the page.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the beginning of the page.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the beginning of the page.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Edit Replace File – End Page

The screenshot shows the 'Edit Replace File' dialog box with the 'EndPage Parameter' tab selected. The 'None' radio button is selected. The 'From file' and 'SmartPrinter File' options have associated text input fields. The 'Plain text' option has a large text area. The 'ASCII sequence' section includes a table with 12 columns and two rows for Hex and Ascii values, along with 'Add' and 'Delete' buttons. The bottom of the dialog features 'Help', 'Cancel', 'Apply', and 'OK' buttons.

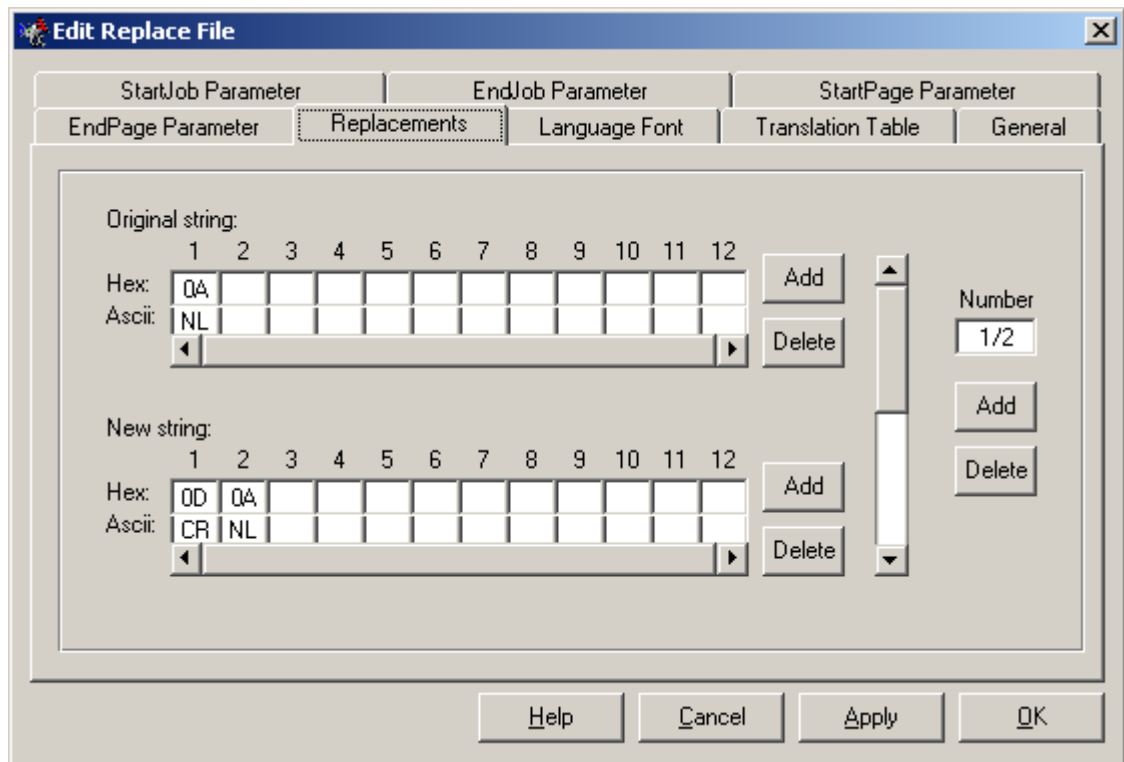
The End Page parameter is a place where you can put information that will be added to the print job at the end of each page. The information can be one of the following types:

- * **None**
No information will be added.
- * **From File**
The file specified will be inserted at the end of the page.
- * **SmartPrinter File**
The [SmartPrinter file](#) (SPF) specified will be inserted at the beginning of the print job.
If the SPF is a public one, it will translate properly on any computer, but if it's a private SPF it will work only on the computer it is intended to.
- * **Plain Text**
The text entered will be printed at the end of the page.
- * **ASCII Sequence**
The ASCII sequence entered will be printed at the end of the page.
Click [here](#) for more instructions on the way to edit the ASCII Sequence.

Edit Replace File – Replacements

The replacements tab is a place where you can make replacement rules for the replace file. A replacement rule is an instruction to the SmartPrinter to replace a specific ASCII string with a different ASCII string.

This ASCII string may be a specific one, but may also use [wildcards](#).



Replacements parameters

* **Original String**

The original string specifies the ASCII string that the SmartPrinter should look for.

Click [here](#) for more instructions on the way to edit the ASCII Sequence.

* **New String**

The new string specifies the ASCII string that the SmartPrinter put instead of the original string.

Click [here](#) for more instructions on the way to edit the ASCII Sequence.

* **Replacement Number**

The replacement number shows you in which replacement rule you are now. You can use the vertical scroll bar to navigate between the replacement rules for this template.

* **<Add> button**

When you click this button, a new replacement rule is added to the replace file. The replacement rule is added before the replacement

rule you were on. Use the [sequence editor](#) to edit the new replacement rule.

* **<Delete> button**

When you click this button, the replacement rule you were on is deleted from the replace file.

Edit Replace File – Language Font

The Language Font tab is a place where you can set font parameters for the replace file. It is usually not recommended to use font from the replace file, since the template identification phase will have problem determining the pages information due to the font binary data. However, if you skip the template identification phase and still need to add font information, here is the place to do it.

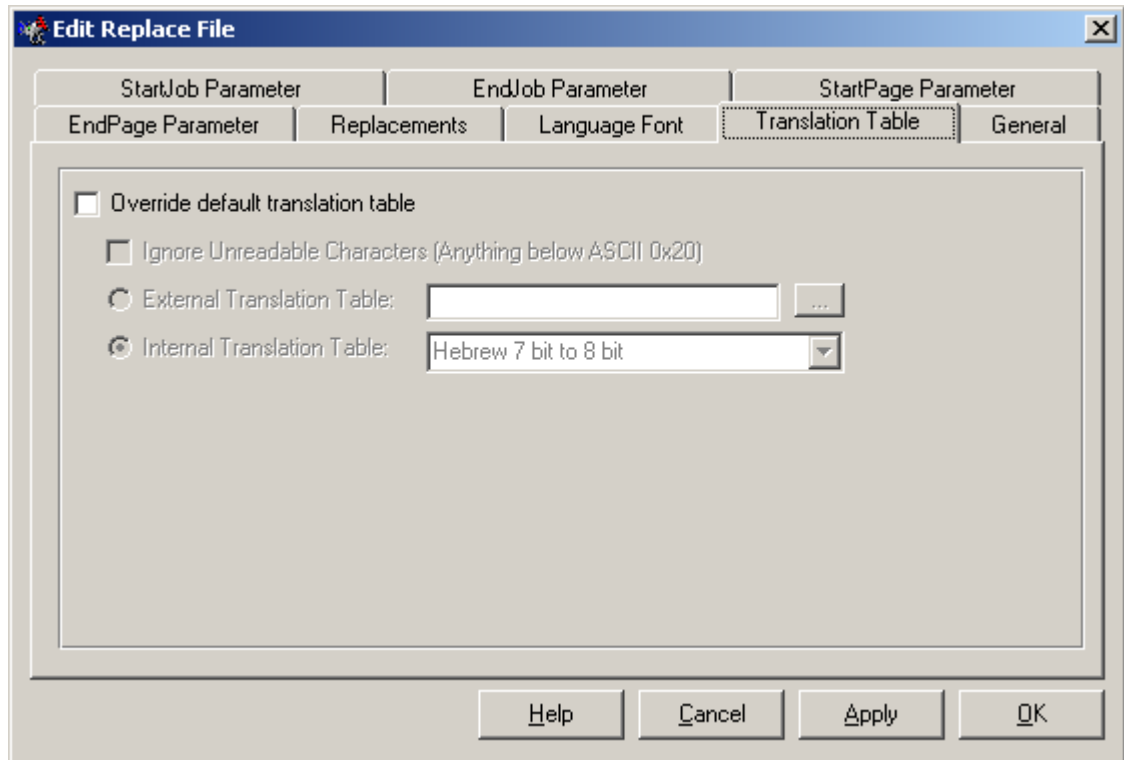
The screenshot shows the 'Edit Replace File' dialog box with the 'Language Font' tab selected. The dialog has several tabs: StartJob Parameter, EndJob Parameter, StartPage Parameter, EndPage Parameter, Replacements, Language Font, Translation Table, and General. The 'Language Font' tab contains a warning message: 'Downloading a font in the replace phase will cause a problem with the key identification phase. It is being skipped!'. Below the warning, there is a checkbox labeled 'Download language font to printer (Replacement phase)' which is checked. Under this checkbox, there are two radio buttons: 'External font file:' and 'Internal font file:'. The 'Internal font file:' radio button is selected, and it is followed by a dropdown menu showing 'David (Hebrew 862)'. Below the radio buttons, there is a section titled 'Internal font properties' which contains several checkboxes and input fields. The checkboxes are: 'Download narrow/wide font' (checked), 'Download proportional font' (unchecked), 'Download Reversed font' (unchecked), 'Download MICR (CMC7) font' (unchecked), 'Replace A-B, C-D' (unchecked), 'Download OCR-A font' (unchecked), 'Download E13B font' (unchecked), and 'Add Initial (default) font size and style:' (checked). The input fields are: 'Style number:' with value '0', 'Symbol set:' with value '15Q', 'Typeface number:' with value '4099', and a text box for 'Add Initial (default) font size and style:' containing the string 's0s0b10.4H'. At the bottom of the dialog, there are four buttons: 'Help', 'Cancel', 'Apply', and 'OK'.

Font parameters

When you check the download language font to printer checkbox, the selected font will be downloaded to the printer at the end of the general replacements phase. The properties of the font are exactly the same as explained in the [Font Management dialog box](#) in the PrintController.

Edit Replace File – Translation Table

The Translation Table tab is a place where you can set translation table for the replace file. The translation table here is useful if you need different translation tables for different printers, in such case use the [replace file manager](#) to associate a replace file with a printer and the translation table of that replace file will override the default translation table.

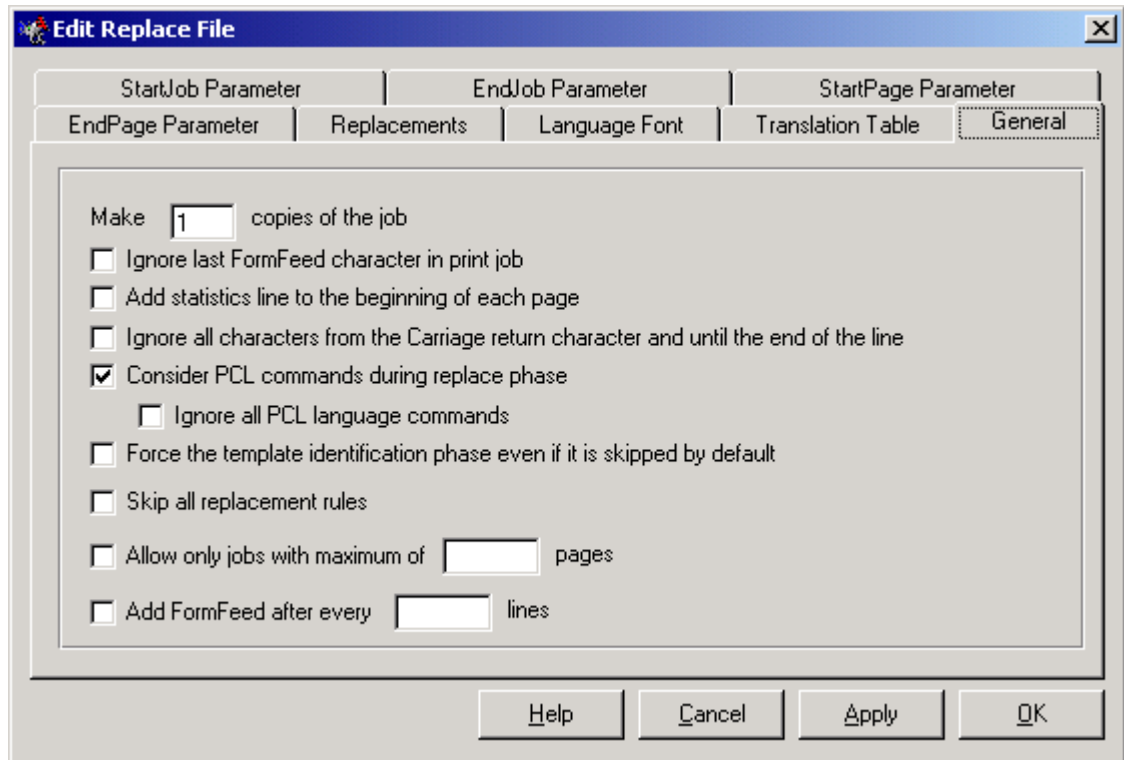


Translation table parameters

When you check the Override default translation table checkbox, the selected translation table will be used for this replace file instead of the default translation table. The properties of the translation table are exactly the same as explained in the [Translation Table Management dialog box](#) in the PrintController.

Edit Replace File – General

The Language Font tab is a place where you can set font parameters for the replace file. It is usually not recommended to use font from the replace file, since the template identification phase will have problem determining the pages information due to the font binary data. However, if you skip the template identification phase and still need to add font information, here is the place to do it.



General parameters

* Job Copies

This parameter allows you to make copies of the entire job. The default value is 1 copy.

* Ignore last FormFeed character in print job

When this checkbox is checked, and the print job ends with a FormFeed character, then this last FormFeed character will be discarded.

* Add statistic line to the beginning of each page

When this checkbox is checked, a special statistic line will be added to the beginning of each page of the print job. The structure of the line is:

JobCopy XX : JobPage YYYY : TotalPage ZZZZ

XX is the number of the current copy of the job.

YYYY is the number of the page in the current job.

ZZZZ is the number of the page in total count

* Ignore all characters from the Carriage return character and until the

end of the line

When this checkbox is checked, the SmartPrinter will ignore any character coming between a CR and LF characters. That is, usually a line ends with a CR-LF characters that will return the writing head to the beginning of the line and advance a line. In some cases, there are repetitions of lines that occur when using only the CR character at the end of the line. This option will allow you to ignore these lines.

*** Consider PCL commands during replace phase**

This parameter handles only the general replacements phase and not the template identification phase. When this checkbox is checked, the SmartPrinter will look for any legal PCL commands. If it is not checked the SmartPrinter will not look for it.

This is very important when a translation table of any type (internal or external) is in use, since then SmartPrinter will not translate any character that is a part of a legal PCL command, if the checkbox is not checked, all characters will be translated including the ones inside PCL commands.

*** Ignore all PCL language commands**

This parameter handles only the general replacements phase and not the template identification phase.

When this checkbox is checked, the SmartPrinter will remove all the PCL command from the print job and will produce a “clean” print job.

The commands removal occurs in the [replace phase](#), it means that in the template identification phase the print job is already cleaned.



The “Ignore all PCL language commands” option can only be used if the “Consider PCL commands during replace phase” option is selected. This is because if the SmartPrinter will not consider the PCL language commands, it will not be able to ignore it.

*** Force the template identification phase even if it is skipped by default**

When this checkbox is checked, the SmartPrinter will do the template identification phase anyway. If the phase was to be done anyway, then this parameter has no effect. However, if the template identification phase was skipped in the [Options dialog box](#), it will be performed anyway.

This is useful when you want only some printers to perform the template identification phase and not all, in this case you skip the template identification phase from the [Options dialog box](#), make different replace file for each printer and only in some of the replace files check this checkbox.

*** Skip all replacement rules**

This parameter handles only the general replacements phase and not the template identification phase.

When this checkbox is checked, the SmartPrinter will do only the start and end job parameters, and will not do replacements themselves. This is useful in places where you want to use an external DLL file to do the work on the file arriving, and you don't want to waste time on searching for replacements in the file (which takes a considerable amount of time).

*** Allow only jobs with maximum of ... pages**

This parameter handles only the general replacements phase and not the template identification phase.

When this checkbox is checked, and a valid value of maximum pages is selected, the SmartPrinter will process job as usual, but at the actual print phase it will check if the number of pages of the current job exceeds the maximum pages allowed, and if so - the job will not be printed. This allows you to define certain printers that will allow to print only small jobs and if you want, you can reprint the cancelled jobs to another printer.

*** Add FormFeed after every ... lines**

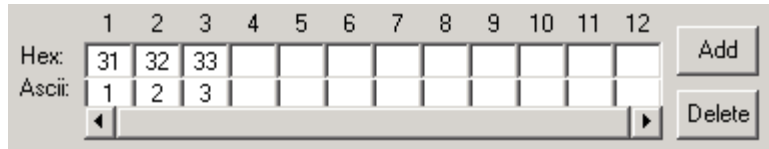
This parameter handles only the general replacements phase and not the template identification phase.

When this checkbox is checked, and a valid value of lines is selected, the SmartPrinter will add a FormFeed Character after the selected amount of lines, causing the job to split long pages into shorter pages with a known

amount of lines.

Sequence Editor

The sequence editor is the tool in which you can edit ASCII strings. It is useful to create ASCII sequences containing unprintable characters such as CR (Hex=0D), LF (Hex=0A), Esc (Hex=1B) and so on.



Sequence editor parts:

- * **Hex row**

In the Hex row, you can see the hex values of the string being edited. When you enter a value to a cell in the Hex row, the appropriate ASCII character that matches the Hex value is printed in the cell below the Hex cell.

- * **ASCII row**

In the ASCII row, you can see the ASCII values of the string being edited. When you enter a value to a cell in the ASCII row, the appropriate Hex value that matches the ASCII character is printed in the cell above the ASCII cell.

- * **Numbering**

The numbering field shows what is your current location in the ASCII sequence being edited. You can use the horizontal scroll bar at the bottom of the sequence editor to navigate in the sequence edited.

- * **<Add> button**

When you click this button, a new character is added to the ASCII sequence. The character is added before the character you were on. The new character added has the Hex value of 0. You need to change that value to the desired character from the hex cell or from the ASCII cell.

- * **<Delete> button**

When you click this button, the character you were on is deleted from the ASCII sequence.

Replacement wildcards

Sometimes you wish to make a replacement rule, but there is a certain character you don't know exactly what it is, or there are many options as to what can it be, and besides these options, nothing else can occur. In these cases you may want to use a wildcard. A wildcard is basically an unknown character. There are two kinds of wildcards:

* General wildcard

The general wildcard is used to replace an unknown character with the same character that is in the same location as the previous one in a certain ASCII sequence.

Example:

The screenshot shows a software interface for string replacement. It has two main sections: 'Original String' and 'New String'. Each section contains a table with 12 columns, numbered 1 to 12. The 'Original String' table has Hex values: 41, 41, 41, ??, ??, 42, 42, 42, and empty cells for 9-12. The corresponding ASCII values are: A, A, A, ??, ??, B, B, B, and empty cells for 9-12. The 'New String' table has Hex values: 43, 43, 43, ??, 44, 45, 45, 45, and empty cells for 9-12. The corresponding ASCII values are: C, C, C, ??, D, E, E, E, and empty cells for 9-12. To the right of each table are 'Add' and 'Delete' buttons. On the far right, there is a 'Number:' label with a box containing '1/3', and another set of 'Add' and 'Delete' buttons. Navigation arrows are present below the tables.

In this example, a string such as AAAYzBBB will be replaced with the string CCCyDBEE.

There are actually 2 ways to use a general wildcard:

1. Source and target (such as letter No. 4 in the example): means that any letter is acceptable to that replace rule in these place and it will be replaced with the same letter in the same place.
2. Source only (such as letter No. 5 in the example): means that any letter is acceptable to that replace rule in these place, but it will be replaced with the letter written in the new string at the same place.



You may use a "Target only wildcard" but it is just as writing the same letter as the letter written in the original string, however it is not an illegal act.

* Specific wildcard

The specific wildcard is used to replace an unknown character with the same character that is in the some other location then the previous one in a certain

ASCII sequence.

Example:

The screenshot shows a software interface for string replacement. It has two main sections: 'Original String' and 'New String'. Each section has a table with 12 columns, numbered 1 to 12. The 'Original String' table shows Hex values: 41, 41, 41, ?1, ?2, ?3, 42, 42, 42, and then three empty cells. The corresponding ASCII values are: A, A, A, ?1, ?2, ?3, B, B, B, and then three empty cells. The 'New String' table shows Hex values: 43, 43, 43, ?3, ?1, 45, 45, 45, and then three empty cells. The corresponding ASCII values are: C, C, C, ?3, ?1, E, E, E, and then three empty cells. To the right of each table are 'Add' and 'Delete' buttons. On the far right, there is a 'Number:' field showing '1/3' and another 'Add' and 'Delete' button. A vertical scrollbar is also present.

In this example, a string such as AAxyzBBB will be replaced with the string CCCzxEEE.

Any numbered wildcard is a specific wildcard, which means that the characters will be replaced by the order in which they were set, and the 1st wildcard will be replaced with the 1st, the 2nd with the 2nd and so on...



You may use a "Source only specific wildcard" but it is just as writing a general wildcard, since any letter will fit there but it will not be used.



You may **NOT** use a "Target only specific wildcard" – such as specifying "?3" in the target output without specifying it in the original string. Doing so may result in unexpected errors.

* **Date/Time wildcard**

This wildcard is used to inject date/time stamp into the ASCII sequence.

- ?D?D - will inject the day
- ?M?M - will inject the month
- ?Y?Y?Y?Y - will inject the year
- ?H?H - will inject the hour
- ?N?N - will inject the minute
- ?S?S - will inject the second



You may combine specific and general wildcards in the same replacement rule as long as you follow the instructions above.

Event Log Viewer

The event log viewer is an important tool to discover and handle abnormal print results. All of the SmartPrinter events, errors, remarks, etc... are written into a log file, which can easily and comfortably be shown in the Event Log Viewer.

	Type	Code	Event	Date	Time	Info
1	Information	30000000	Job printed successfully:	1/1/2004	12:36:04	Printer: lumen
2	Information	30000000	Job printed successfully:	1/1/2004	12:37:15	Printer: lumen
3	Information	30000000	Job printed successfully:	1/1/2004	12:38:49	Printer: lumen
4	Information	30000000	Job printed successfully:	1/1/2004	12:40:16	Printer: lumen
5	Information	30000000	Job printed successfully:	1/1/2004	13:08:39	Printer: gest89
6	Information	30000000	Job printed successfully:	1/1/2004	13:09:18	Printer: gest89
7	Information	30000000	Job printed successfully:	1/1/2004	13:11:59	Printer: gest89
8	Information	30000000	Job printed successfully:	1/1/2004	13:13:36	Printer: gest89
9	Information	30000000	Job printed successfully:	1/1/2004	13:17:06	Printer: gest89
10	Information	30000000	Job printed successfully:	1/1/2004	13:19:32	Printer: gest89
11	Information	30000000	Job printed successfully:	1/1/2004	13:24:32	Printer: gest89
12	Error	10000000	PCL Command two long,	1/1/2004	13:25:41	PCL Command may
13	Information	30000000	Job printed successfully:	1/1/2004	13:26:35	Printer: gest89
14	Information	30000000	Job printed successfully:	1/1/2004	14:09:56	Printer: gest89
15	Information	30000000	Job printed successfully:	1/1/2004	14:10:57	Printer: gest89
16	Error	10000000	PCL Command two long,	1/1/2004	14:11:10	PCL Command may
17	Error	10000000	PCL Command two long,	1/1/2004	14:17:14	PCL Command may
18	Information	30000000	Job printed successfully:	1/1/2004	14:18:52	Printer: gest89
19	Information	30000000	Job printed successfully:	1/1/2004	14:20:28	Printer: gest89

Event Log Operations:

<Clear> This button is used to erase the entire event log.

<Export to File...> This button is used to export the events in the event log into Microsoft Excel File.

<Event Types...> This button is used to select which events will be displayed in the log, and which will be ignored.

<Refresh> This button re-reads the event log file. If the event log is opened, and more events are written to the event log, it will not show on the Event Log Viewer until the refresh button will be clicked.

<Close> This closes the Event Log Viewer.

<Help> This button shows this help file.

Event Information:

For each event in the log file, the following information is available:

Type: Each event has its own type. The type may be one of the types listed below.

Code: Each event has its event code. The event code helps identifying the event in the process of analyzing the problem.

Description: This field contains the description of the event.







Date: The date in which the event has occurred.

Time: The time in which the event has occurred.

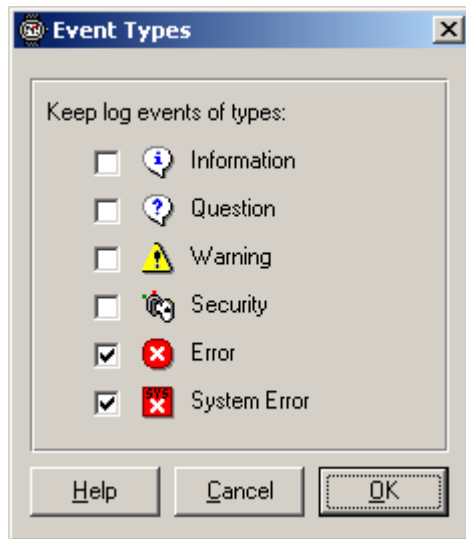
Information: This field contains more relevant information related to the event, in addition to the information supplied in the description field.

Event Types:

There are several types of events that the SmartPrinter reports about:

-  **Error:** When an event of type error appears, it indicates that an error that interrupted the print process has appeared. Usually after an error has occurred, the print output will be either corrupted or missing.
-  **Warning:** When an event of type warning appears, it indicates that there was some problem during the print process, but the problem was not severe enough to interrupt the print process. Nevertheless, the print output may not be what you intended it to be.
-  **Information:** When an event of type information appears, it usually indicates that a correct and valid operation was performed.
-  **Security:** When an event of type security appears, it indicates that a security problem has appeared, either with the SmartPrinter license or with a secure print process.
-  **System Error:** When an event of type system error appears, it indicates that a severe system error (an error which came from the windows operating system) that interrupted the print process has appeared. Like regular error, usually after a system error has occurred, the print output will be either corrupted or missing.
-  **Question:** When an event of type question appears, it indicates that there are some illogical or strange parameters in the system and you should check these parameters soon. However these parameters do not disturb the regular print process since no error has appeared.

Select Event Types



Using the event types dialog box, you can decide which event types will be written to the event log and which event types will not be written to the event log. Each of the checkboxes in this dialog box handles an event type, when you check a checkbox, the suitable event type will be enabled for event log writing. Click [here](#) for more information on the event log and the event types.

If you wish to disable all event types from being written to the event log, simply uncheck all of the checkboxes.

Export to File...

You may want to export the log file into a Microsoft Excel file. This is a comfortable way to handle your log file, make statistics and analyze the events.

In order to export the log file, all you have to do is:

1. Click <Export to File...>.
2. Select a name for the Excel file, and click <Export>.

The file created will be a standard Microsoft Excel file.