Adobe™

UNIX Daemon Administration Guide

Adobe Acrobat version 3.0

Contents

Adobe Acrobat 3.0

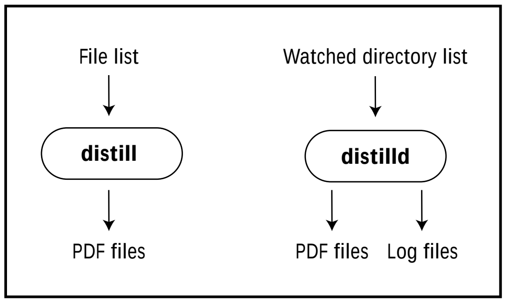
UNIX Daemon Administration

The Distiller daemon monitors designated directories for PostScript files to be distilled into PDF files. When it finds these files, it distills them. The designated directories the daemon monitors are called *watched* directories. This guide explains how the Distiller daemon works and the steps needed to configure and maintain one or more daemon processes.

You must install the Acrobat Distiller to use the Distiller daemon. See the *Acrobat 3.0 Installation Guide* for directions on installing the Distiller daemon software.

Overview

The Distiller daemon (`distilld`) is essentially an interface to the Distiller (`distill`). You provide Distiller with the names of one or more files to distill. When you run the Distiller daemon, you provide it with a list of watched directories.



How the Distiller and Daemon differ

When you invoke the Distiller from the command line, it processes only the files listed on the command line. When you invoke the Distiller via the daemon, it processes all files placed in watched directories by any user.

Starting the daemon

You can run a daemon either from the command line or from a system startup file. In either case, the command line that starts the daemon is the same.

Command line

The easiest way to run the daemon is with no command line options specified. If you start the daemon in this simple way:

```
distilld -dirs [names of watched directories]
```

text similar to the following is output before returning to the shell prompt.

```
Acrobat Distiller Daemon 3.0 for SunOS/Solaris (SPARC)
Daemon process ID: 1557
System config file:
/tmp/mars_nov21_1557.config
System log file:
/tmp/mars_nov21_1557.log
```

The first line gives the version number and platform of the daemon executable. The second line gives the process ID of the daemon. The third and fourth lines give the location and filename for the system configuration and log files.

All further messages from the daemon will be logged to the system log file. If the daemon is unable to write to this file, it sends a message to the console and exits.

The Distiller daemon is now running to monitor the directories. You can later change the distillation parameters for a particular directory (`distilld -configonly` on that directory) and any running daemon monitoring that directory will notice the change. Changes to the basic distillation parameters or monitoring parameters will affect only subsequently started daemons. You must kill and restart the daemon for these parameters to take effect.

Startup script

You can use the following sample code to invoke a daemon from a system startup file. Use it as the starting point for the code you add to your startup file.

```
#start a distiller daemon to monitor watched
#directories
#
if [ -x <bindir>/distilld ]; then
   <bindir>/distilld > /dev/null
   echo -n ’distilld’
fi
```

Replace <***bindir***> with the pathname of the directory in which the daemon executable was installed. The ***distilld*** command immediately forks a background process (which is the daemon) and then terminates.

This is the suggested way to invoke the daemon from a system startup file. You can specify ***-nodirprefs*** and ***-noparamprefs*** to avoid reading any existing user preferences file.

Each daemon subsequently started contains an echo statement resulting in a message to standard out something like:

```
 starting local daemons:sendmail nfsd statd lockd distilld
```

When distilld (the daemon process) starts, it outputs to standard out the version number and platform of the daemon executable, the process ID of the forked daemon process, and the filenames for the system configuration and log files. Normally, you should direct this output to ***/dev/null***.

Configuring

When invoked, the daemon determines which directories are monitored, what distillation parameters are used, and what is to be written in log and configuration files. If the daemon is running in normal mode, it determines its configuration and then begins to monitor directories and distill files. If it is running in configuration mode, it determines its configuration and then quits.

The default mode for the daemon is normal operating mode. There is no option to specify this mode. In normal mode, the daemon runs in the background and performs all activities.

The ***-configonly*** option runs the daemon in configuration mode. In configuration mode, the daemon runs in the foreground and performs only configuration activities. Configuration mode is used to initialize parameter settings or change existing ones.

Parameters

There are several categories of parameters:

- Some of the parameters affect what the daemon does while monitoring the watched directories.
- One of the parameters designates the beginning of the list of watched directories.
- The rest of the parameters specify distillation parameters that a Distiller process can also use when distilling files.

There are two groups of parameters: general parameters and directory-specific parameters. General parameters apply to the operation of the monitoring daemon and apply in all the directories the daemon monitors. The directory-specific parameters apply to distillation activities in a specific directory.

The following table lists the daemon configuration options and parameters. When you specify these parameters on the command line, you must precede each one with a dash (**-**). Some of the parameters may also take arguments.

Table 1: *Distiller Daemon Configuration Options and Parameters*

distilld	**CONFIGURING**			
	configonly	saveparamprefs	noparamprefs	newdprefs
		savedirprefs	nodirprefs	
	MONITORING PARAMETERS			
	pause	srcdisp	sysloglimit	
	resume	destexp		
	DIRECTORY LIST PARAMETERS			
	dirs			
distill/distilld	**DISTILLING PARAMETERS**			
	General	**Compression**	**Font Embedding**	**Advanced**
	compatlevel	monores	subsetfonts	includebookends
	asciipdf	grayres	maxsubsetpct	preserveopi
	pagesize	colorres	alwaysembed	preserveoverprint
	resolution	colordepth	neverembed	preservehalftone
		monodepth	embedallfonts	transferfunction
		graydepth		cmyimagestorgb
		colorcompr		UCRandBG
		graycompr		colorconversion
		monocompr		
		compresstext		
		colordownsample		
		graydownsample		
		monodownsample		
		colordownsampletype		
		graydownsampletype		
		monodownsampletype		
		encodecolor		
		encodegray		
		encodemono		
		coloracs		
		grayacs		
		coloracsq		
		grayacsq		

You can also get information about these parameters by running ***distilld*** with the ***-help*** option.

Parameter sources

The Distiller daemon parameter list is built, in a specific order, from a number of different sources.

The sources are listed in the table below in the order in which they are read. Each source overrides any values set by previously read sources. If a cell contains a checkmark, that parameter source can provide settings for parameters in that category.

Table 2: *Sources of Options and Parameters for the Daemon*

Source and Order		Parameters			Options	When/How Used
		Distilling	Monitoring	Directories	Configuring	
1	Application Defaults	✔	✔			Always read. Provides defaults for all the distillation and monitoring parameters needed by the daemon.
2	PostScript startup files	✔				Always read if they exist. (Optional) Provides only distillation parameters, and only may set a few of them.
3	User Preferences file	✔	✔	✔		Read by default. Ignore with the ***-noparamprefs*** and ***-nodirprefs*** options. (Optional) Provides distillation and monitoring parameters as well as a list of watched directories.
4	DPREFS file	✔				Read by default. Ignore with ***-newdprefs*** option. Automatically created the first time distilld monitors a directory. It provides only distillation parameters.
5	Command line	✔	✔	✔	✔	Always read. Provides any parameters. The only place to specify configuration options.

You can start a daemon with no parameters on the command line if a list of directories exists in the user's preferences file. If no such list exists, you must at a minimum specify the list of directories on the command line. When the daemon prepares to distill a file in a watched directory, it retrieves its basic configuration from memory and overrides some of the parameter values with values specified just for that watched directory. This allows you to set up directories with different distillation parameters and have the same daemon watch all of them. This override is directory specific; it is done on the fly and does not overwrite the basic configuration information in memory. See "Parameters" on page 14 for more information.

The daemon reads the first three sources when it initializes and stores the resulting parameter list in memory. It reads the fourth source each time it enters a watched directory. This source can be thought of as a per directory override to the initial configuration. After applying these overrides, parameters that were specified on the command line are applied. The resulting parameter list is the one that is used when it distills a file in that directory. If this list differs from what is currently recorded in the DPREFS file, the daemon rewrites the DPREFS file with the list of distillation parameters it just used.

User preferences file

The user preferences file, ***$HOME/AcrobatDistiller***, stores some of the parameter values used during a particular invocation of the daemon. These parameter values can also be used by subsequently invoked daemons. The values specified in this file override any defaults, and can in turn be overridden by values specified on the command line.

When the daemon is invoked with the ***-saveparamprefs*** option and one or more distillation parameters, the user preferences file records all the parameters. If you cause only the preferences file to be written from a command line with monitoring parameters and configuring options, but no distillation parameters, the distillation parameters will not be written (or overwritten if they already exist) in the preferences file.

Any user who runs a daemon with the appropriate configuration options can cause the daemon to create or modify their user preferences file. See “Saving user preferences” on page 16 for additional information.

DPREFS files

When you run the daemon to configure a specific directory or list of directories, it automatically writes the distillation parameters into the DPREFS file in the watched directory. The DPREFS file records the distillation parameters used to distill the last file in that directory. Each watched directory has a DPREFS file. The file is created the first time a daemon is configured, or the first time a daemon monitors the watched directory.

When the daemon enters the directory and finds a file to distill, it retrieves the basic configuration, overrides it with the parameter values in the DPREFS file, and overrides that result with any values specified on the command line. The resulting list of distillation parameters is used to distill the next file in that directory. The list of distillation parameters is then compared with those in the DPREFS file; if there is a difference, it writes the list it just used into the DPREFS file, overwriting the previous contents. Since this file is generated by a union of the basic configuration and overrides, it also provides a complete list of the distillation parameters.

Creating and registering watched directories

There are only two situations where you need to begin the list with the ***-dirs*** parameter. One is when you need to separate the directory list from a font list given as an argument to the ***-neverembed*** or ***-alwaysembed*** parameter. The other is when the first directory name in the watched directory list begins with a dash (-). List directories as the last arguments.

Define the directories into which your users will place files to be distilled. You can use existing directories or create new ones for this use. Once the directories are created, you must do two things: 1) provide the monitoring daemon with the directory names, and 2) set the appropriate permissions on the directories and their subdirectories so the daemon and the intended users have the necessary access.

If you set up a number of watched directories, the permissions on each may vary, depending on the intended use. You may have a variety of users or user groups, each using only one or two of the directories. You should define the expected use of each watched directory and identify the users that will use it in order to define appropriate groups and set the correct permissions on the directories.

Define watched directories

The daemon needs a list of directories to watch. One way to do this is to run the daemon in configuration mode, specify the list of directories, and save them in the user preferences file. The following code shows how to do this:

```
distilld -configonly -nodirprefs -savedirprefs dir1 dir2 dir3 . . .
```

The ***-nodirprefs*** option prevents the daemon from reading any existing directory names from the preferences file. The list will be only those specified on this command line. The ***-savedirprefs*** causes the list you've specified to be saved in the user preferences file, overwriting any other directory list.

Alternatively, you can specify the list of directories you want the daemon to monitor when you invoke the daemon, avoiding use of the user preferences file altogether:

```
distilld -nodirprefs . . . dir1 dir2 dir3 . . .
```

Example: Record a list of watched directories

The following example specifies a list of directories to be watched and records the list in the user preferences file for possible use by subsequently invoked daemons.

```
distilld -configonly -nodirprefs -savedirprefs \
   /proj/distiller/doc /proj/exchange/doc \
   /proj/maker/doc
```

The ***-savedirprefs*** option causes the daemon to save the list of directories to be monitored in the user preferences file (for whichever user runs the command). The ***-nodirprefs*** option causes the daemon to ignore any existing directory list in the user preferences file. Three directories are recorded in the watched directory list.

Example: Add to the list of watched directories

This example adds two more directories:

```
distilld -configonly -savedirprefs \
   /proj/makersgml/doc /proj/photoshop/doc
```

The absence of the ***-nodirprefs*** option causes the daemon to read the list of directories already in the user preferences file. It then adds the two directories specified on the command line to this list. The ***-savedirprefs*** option causes the daemon to write the resulting list to the user preferences file.

Set directory permissions for the daemon

The daemon needs read and write access to each watched directory and its *IN* and *OUT* subdirectories. It needs write permission to the watched directory to create the *IN* and *OUT* subdirectories as needed, and to create and write the *DPREFS* and *DTIME.TXT* files.

The daemon needs full access to the *IN* directory so it can scan it for PostScript language files and remove the files after they're distilled. It needs full access to the *OUT* directory so it can create and write PDF files and move PostScript files into it. It may also need to remove files from the *OUT* directory.

The daemon runs with the permissions of the user who invoked it. To give the daemon both read and write permission to a directory, the user who invoked it must have read and write permission to the directory.

Set directory permissions for the users

If you have more than one user using a watched directory, define a group to which those users belong, and then set the appropriate permissions on the directories for the group.

Users put their PostScript files into the *IN* directory of a watched directory and retrieve the distilled PDF files from the corresponding *OUT* directory. Minimally, they need read/write/execute permission on the watched directory, read/write/execute permission on the *IN* directory, and write/execute permission on the *OUT* directory.

Define a "private" watched directory

Users may want to distill files without placing them in publicly readable directories. A "private" directory can be watched by defining a directory that gives read/write permissions only to a single user:

```
chmod u+rwx og-rwx <dir>
```

Configuration examples

The following examples show a few of the many possible ways you can configure the daemon and its directories. Use one or more of these as starting points to achieve the desired configuration at your site. In all of these examples (unless otherwise noted) the first option to distilld is ***-configonly.*** This causes the daemon to do configuring activities and then terminate.

You can run a daemon process in configuration mode while one or more other daemon processes are running to monitor watched directories. Changes made by the daemon running in configuration mode to the *DPREFS* file in a watched directory will be automatically picked up the next time a daemon monitors the directory. Configuration changes recorded in the user preferences file will not be picked up by currently running daemons.

Example: General configuration

```
distilld -configonly -nodirprefs -saveparamprefs \
   -lzwtext on -colorcompr jpeg:medium \
   -srcdisp movetoout -destexp 15
```

After the daemon has determined configuration, monitoring and distilling parameter settings are recorded in the user preferences file (***-saveparamprefs***).

In this example, the daemon doesn't have a list of directories to configure (***-nodirprefs*** and no directories specified on the command line). It configures only the general parameters and saves them in the user preferences file. It doesn't write a list of directories to that file and it doesn't regenerate any *DPREFS* files.

The daemon sets several distillation parameters (***-lzwtext*** and ***-colorcompr***). A subsequent daemon that reads the user preferences file upon initialization will move the original PostScript files from the *IN* to the *OUT* subdirectory when it is finished distilling them (***-srcdisp***). It will also remove files from the *OUT* directory that are older than 15 days.

Example: Override defaults for specified directories

This example sets some of the distillation parameters to be used for two directories: */proj/excalibur/doc* and */disks/pcserver1/monitored*. Any daemons monitoring these directories will pick up these values from the *DPREFS* file there. Unless a direct command line override was specified when the currently running daemon was invoked, this configuration change will affect any daemons currently running (the next time they distill a file from that directory) as well as any subsequently started.

```
distilld -configonly \
   -colorcompr jpeg:1.0 -graycompr lzw \
   -nodirprefs /proj/excalibur/doc \
   /disks/pcserver1/monitored
```

Several distillation parameters are set here (***-colorcompr*** and ***-graycompr),*** and ***-nodirprefs*** prevents the daemon from working on any other directories than the two specified on the command line. If this option were not specified, the daemon would configure all the directories listed in the user preferences file in addition to the two specified on the command line.

Since the ***-saveparamprefs*** option is not specified, the configuration change is not recorded in the user preferences file. However, the *DPREFS* files in the two specified directories are rewritten with the new values for the distillation parameters.

Monitoring and distilling

The following section describes the process of monitoring and distilling files in watched directories.

Monitoring process

The daemon monitors its list of directories in a round-robin fashion. If there are 10 PostScript files in each watched directory, it will distill one file from each directory before returning to the first directory to make a second sweep. This method ensures that the daemon does not neglect other directories because it's been busy working in one directory that is very full, has large files that take a long time to distill, or is heavily used.

Each watched directory has an *IN* and an *OUT* subdirectory. PostScript files placed in the *IN* directory are distilled into corresponding PDF files in the *OUT* directory. When the daemon is finished distilling a file, it (optionally) moves the PostScript source file from the *IN* directory to the *OUT* directory or deletes it.

The daemon also performs monitoring activities including:

- removing old files in the *OUT* directories
- preparing new directories for monitoring
- suspending directories in which it encounters problems
- unsuspending suspended directories
- preparing directories that have recently been unsuspended
- checking directories that recently had temporary problems

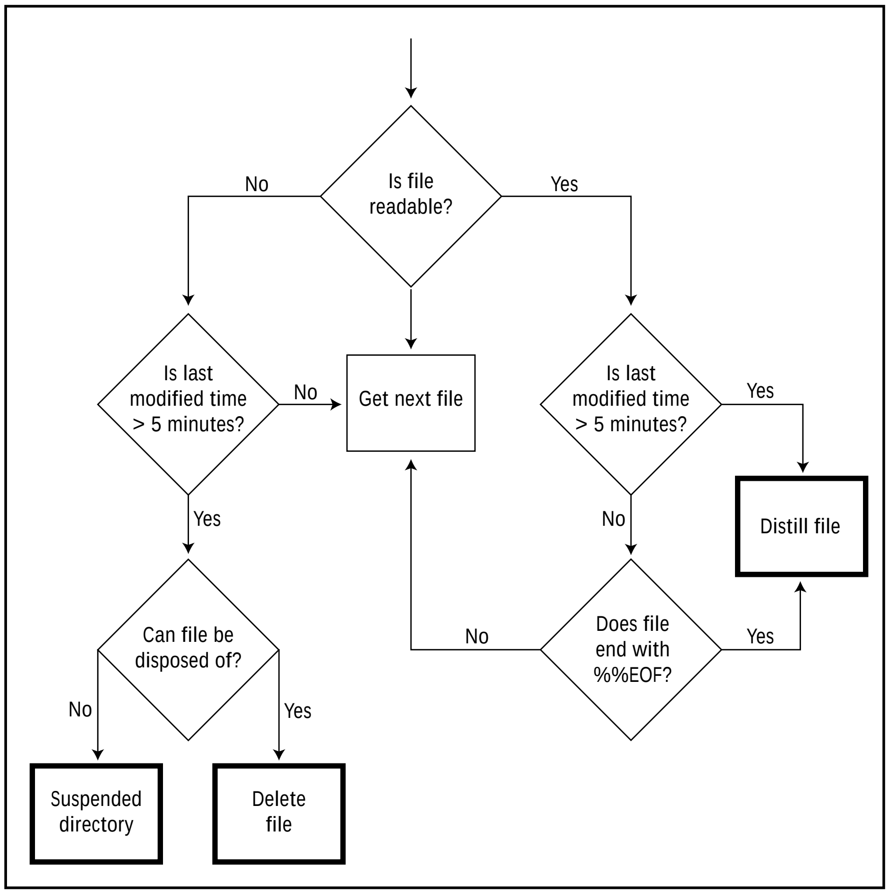
Suspending directories

If the daemon can't gain appropriate access to a monitored directory, it suspends monitoring of that directory. When a watched directory is suspended, the daemon stops monitoring it. This prevents the daemon from repeating attempts to access an inaccessible directory or distill a file with problems.

For example, the daemon will suspend a watched directory if the *IN* subdirectory of the watched directory suddenly becomes unreadable or unwritable. If it becomes unreadable, the daemon cannot access the files to be distilled. If it becomes unwritable, the daemon would distill a file in the watched directory and then be unable to remove the file from the *IN* directory. This would cause the file to be distilled again each time the watched directory was monitored, with no PDF file resulting. Similarly, if the *OUT* subdirectory were to become unwritable, the daemon could not write the distillation results to a PDF file.

Determining if a file is ready to be distilled

The daemon checks to determine whether it should pass over a file, dispose of a file, or suspend the directory. The following diagram shows what the daemon does when it enters a monitored directory and finds a PostScript file. Diamonds indicate decision points. Rectangles indicate actions that are performed. Rectangles with heavy black borders indicate the final action performed.



How the daemon determines if a file can be distilled

The daemon determines if a file in a watched directory is:

- a readable file
- an exclusively accessible file
- a PostScript file

Readable? When the daemon finds a PostScript file in an *IN* directory, it determines whether it can read the file. If the invoking user does not own the file or does not belong to the group that has read access, and the "other" permissions do not grant read access, the daemon will be denied read access to the file.

Exclusive Access? The daemon determines if it has exclusive access to the file; that is, no other application is using the file.

The daemon may operate via NFS mounts on files in watched directories that reside on non-UNIX systems. Users on various platforms can create, move, or copy PostScript files into these directories using whatever application or method they wish. Since the daemon can't assume that files are being produced exclusively, the last modified time of the file is used to determine if the daemon has exclusive access. If the file hasn't been modified in the last five minutes, the daemon assumes that no other application or program is using it.

A PostScript file? The daemon checks to determine if the file ends with %%EOF, a PostScript Document Structuring Convention.

Parameters

The following section describes the monitoring, configuration, and distillation parameters.

Monitoring parameters

These parameters determine the behavior of the daemon as it monitors watched directories. These parameters can be saved in the user preferences file and used for future invocations of daemon processes.

File cleanup

-srcdisp — controls what the daemon does with a PostScript file after it has been distilled.
This parameter takes one argument, either ***delete*** or ***movetoout.***

delete — deletes the PostScript file from the *IN* directory
movetoout — (default) moves the PostScript file from the *IN* directory to the corresponding *OUT* directory.

-sysloglimit — controls the size of this log file. This parameter takes one integer argument that specifies the maximum file size for the log. The size is specified in bytes and can be any value greater than 12000. You can turn off system log pruning by specifying a value of -1 for this parameter.

Time intervals

-pause — pauses the daemon for a specified time after it finds all the directories empty. This parameter takes one integer argument between 1 and 10000 and determines the number of seconds the daemon waits after finding all of its watched *IN* directories empty before it begins to check them again. The default value for this parameter is 10 seconds.

-destexp — removes files left in the *OUT* directory after a specified period of time. The parameter takes one integer argument that specifies the number of days the files should stay in the directory before they are removed. You can specify any number of days between 1 and 365. For example, ***-destexp 60*** causes the daemon to remove files more than 60 days old from the *OUT* directory. The default is -1, which causes the daemon to leave files in the *OUT* directory, no matter how long they've been there.

Unsuspending directories

-resume — reactivates suspended directories automatically at fixed time intervals. An argument of ***-1*** disables automatic resumption. A value of ***1*** through ***24*** indicates the number of hours the daemon should wait before unsuspending any suspended directories. The default is 24 hours.

Configuration options

The daemon does *not* save these options in a file for future invocations of the daemon. These options must be specified on the command line when the daemon is started, and apply only to that invocation of the daemon.

These options cannot be changed while the daemon is running. If you want to change one or more of them, you must kill the daemon and restart it. The default setting for all of these options is off (not specified on the command line). To turn them on, you must specify the option on the command line.

Configuration mode

-configonly — causes the daemon to run in configuration mode. The daemon constructs and possibly saves configuration information, but does not monitor directories and distill files. It terminates after its configuration activities are done. To display the default parameters for your current setup, invoke a daemon with only the ***-configonly*** option.

Determining parameter sources

-noparamprefs — ignore any parameter settings saved in your user preferences file. Use this option to override the default behavior, which is to read the user preferences file.

-nodirprefs — don't use the names of any watched directories that are stored in your user preferences file when constructing the list of watched directories. The list of watched directories will be only those specified on the command line.

-newdprefs — ignore the *DPREFS* file for each watched directory it processes. When constructing the distillation parameter list for each watched directory, don't use any settings saved in the *DPREFS* file for that directory.

Saving user preferences

-saveparamprefs — save the current logging, monitoring, and distillation parameters in your user preferences file located in your home directory. These parameters are read by default whenever you start a daemon process. If the daemon is started automatically at system boot time, the root user's preferences file is read. No error occurs if a user does not have a user preferences file.

-savedirprefs — saves the current list of watched directories in your user preferences file. If you specify three directories on the command line and there are four other directories already listed in the user preferences file, the daemon writes the union of these two lists to the user preferences file as the new directory list.

If you want to replace the list currently in the user preferences file, you can use the ***-nodirprefs*** option with the ***-savedirprefs*** option. The daemon replaces the current preferences list with the one you specify on the command line.

The ***-saveparamprefs*** and ***-savedirprefs*** options can be used together.

These options force the recording of the parameters in the user preferences file. They don't influence how the daemon constructs its parameter list for that invocation. They can be used in conjunction with the ***-nodirprefs*** and ***-noparamprefs*** options. If all four of them are specified, the daemon ignores the user preferences file when constructing the parameter list, but records the result into the user preferences file.

Distillation parameters

There are many parameters that control how a file is distilled. They are explained in detail in the Online Guide. These parameters are set by the application defaults, by any PostScript startup files, by the user preferences file, by the DPREFS file in each watched directory, or on the command line. You can run a daemon in configuration mode to modify the parameter settings in the user preferences and the DPREFS files.

Logging

When the daemon determines its configuration, it builds a basic parameter list and then augments that list with directory-specific parameters and command line overrides. It records log information in a similar way by recording application-specific information in two system log files, and writing directory and file-specific information into log files in each watched directory.

System log files

The daemon keeps a record of all its activities. It stores configuration information and all changes to that configuration, the results of its normal monitoring and distilling activities, and any errors. Two log files are used for this purpose: the system log file and the system configuration file.

The system log file

Daemon activities are recorded in the system log file. There is one system log file per daemon. If multiple daemons are run on the same system, there will be a system log file for each daemon.

The system log file contains the following information:

• *the initial configuration of the system* (parameter values) resulting from the initialization of the daemon (the basic configuration).

• *information generated while files are being distilled*. Information normally written to standard out or standard error by an independent Distiller process is written to this file by the daemon.

• *information resulting when the daemon has trouble* with a monitored directory and when it succeeds in recovering from these problems. For example, if the daemon encounters a corrupted DPREFS file and suspends the directory, that fact is recorded in this file. When the directory is later resumed, that fact is also recorded here.

• *configuration changes* caused by a DPREFS file being rewritten. In this case, the configuration change is recorded in the log file, as well as the appropriate portion of the configuration file.

By default, this file is not pruned in size and will never lose data unless the ***-sysloglimit*** parameter was specified in the user's preferences file or on the command line.

The default name of the log file is *hostname_date_daemonPID.log* where *hostname* is the hostname of the system on which the daemon is running, *date* is the current date expressed as month and day, and *daemonPID* is the process ID of the daemon process that is writing the file. For example:

```
neon_jan12_18192.log
```

The default location for this file is determined at install time. The suggested default value is */tmp*. Both the name and the location of this file are configurable. See "Customizing the system filenames and locations" on page 18 for more information.

The system configuration file

The system configuration file is a log file that records the current configuration of the system. All the information is also recorded in the system log file, unless the log file has been pruned. However, the configuration information may be scattered throughout the file.

The configuration information is duplicated here for the convenience of the users and administrators of the daemon setup.

This file records the values used by the daemon for all the parameters discussed in Table 1, “Distiller Daemon Configuration Options and Parameters,” on page 6. It contains the list of directories the daemon is monitoring and a section for each directory that lists the distillation parameters specified for that directory (from the DPREFS file). It also lists the distillation parameters obtained at initialization time and any other parameters set for the daemon. It also contains information about any problems the daemon is having monitoring the directories.

This file is created by the daemon when it initializes. It is modified whenever a DPREFS file is created or rewritten. It is overwritten completely whenever the daemon receives a HUP signal.

The default name of this file is

hostname_date_daemonPID`.config`

where *hostname* is the hostname of the system on which the daemon is running, *date* is the current date expressed as month and day, and *daemonPID* is the process ID of the daemon process that is writing the file. For example:

neon_jan12_18192.config

The default location for this file is determined by the value specified when the daemon was installed. The default suggested at install time is */tmp*. Both the name and the location of this file can be customized. See the next section, “Customizing the system filenames and locations” for more information.

Customizing the system filenames and locations

Setting system configuration and log filenames

The default names for the system configuration and log filenames are determined by resources defined in a UNIX PostScript Resource (UPR) database file, ***Distill.upr***, located in *<installdir>/Distillr/Xtras*. This file is used by both the Distiller and the daemon.

There is one category, ***DstlrRes***, defined in this file. The category includes several resources. ***DstlrSysConfig*** defines the file name or file name pattern for the system configuration file. ***DstlrSysLog*** defines the file name or file name pattern for the system log file.

Setting the filenames

The default setting for these files are:

```
DstlrSysLog=/tmp/\%H_\%D_\%P.log
DstlrSysConfig=/tmp/\%H_\%D_\%P.config
```

In these settings, the %H placeholder evaluates to the hostname of the system, the %D placeholder evaluates to the date in short month/day format, and the %P placeholder evaluates to the process ID of the daemon that creates the file. The “%” in these placeholders must be escaped with “\” to prevent the UPR parsing routines from interpreting them as comment delimiters. This format results in file names like *neon_jan12_19182.log*.

Changing the default names:

• If you plan to run only one monitoring daemon on your system, you can omit any of the placeholders in the filename, or omit all of them and name the file by a static name.

• If you plan to run multiple daemons on the same host, you should use the process ID in the filename. The date may also be useful.

• If you plan to run multiple daemons on different hosts that all use the same (mounted) directory for their log and configuration files, use the process ID and the hostname in the filename specification. The date may also be useful.

• If you plan to periodically save the log file, omit the date specification as part of the filename, since the date is the date on which the daemon process was invoked, not the date the file is first created. In this case, the date will not help differentiate the various archived versions of the log file. Your archiving scheme should use some meaningful naming convention to help you order the archived versions.

Setting the directory

If you wish to change the default names or locations of the system configuration and log files, modify the *Distill.upr* file.

The daemon must be able to write to these files during its operation. Some file systems that provide an NFS interface will refuse clients write access to a file even when the client satisfies the permission requirements. For example, a Novell file system often refuses file access to an NFS client when a non-NFS client has the file open, even just for reading. Do *not* locate the system files on such a system. It will prevent daemon access to system files at indeterminate times and the daemon will exit. The directory in which the system log files are kept must be writable by any user allowed to do administrative tasks for the daemon.

Directory-specific log files

DTIME.TXT files

The *DTIME.TXT* file is an ASCII text file. The contents of this file enable you to determine which daemon is monitoring the directory and what parameters the daemon used to distill the files in this directory.

Every time a daemon distills or tries to distill a file in the watched directory, it makes an entry in this file. Each entry contains:

- job parameters — used to distill the file
- job settings — the name of the system configuration and log files
- time stamp — the last time the distiller daemon checked this watched directory

The first time the daemon monitors a watched directory, it creates a *DTIME.TXT* file in the *OUT* subdirectory. Each subsequent time the daemon monitors the directory, it updates this file.

Running multiple daemons

Multiple daemons can be run on a single host or on multiple hosts. In either case, the daemons can access directories that are on the local system or mounted from a remote system.

Assigning watched directories

There are several ways, which may be used in combination, to assign watched directories to multiple daemons:

- Assign a completely different set of directories to each daemon.
- Define overlapping sets of directories, where two daemons share some directories and monitor others by themselves.
- Define a set of directories that are shared completely by multiple daemons.

Managing file access

If distributed daemons monitor the same directory and the daemons are not all running on UNIX platforms, access to the shared directory must be synchronized so they don't try to distill a file already being distilled by another daemon.

- To handle this situation, a daemon creates a lock file in the *OUT* directory for a file it intends to distill. The name of the lock file is the name of the PostScript source file with the extension "*.lck*". This lock file contains text that explains its purpose and indicates which daemon created it. If the daemon terminates cleanly (for instance, if it receives a ***TERM*** signal), it removes lock files that it created. However, if the daemon crashes or is terminated uncleanly (for instance, by a ***KILL*** signal), it does not remove the pending lock file. In this case, manually remove the lock file before any other daemon can attempt to distill the corresponding PostScript file.

Multiple daemons on a single host

If you have a host with multiple processors, running multiple daemons can make use of this added processing power. Because daemons on the same host (all started by the same user, usually root) read the same user preferences file, you should not record the directory list in the user preferences file (don't use the ***-savedirprefs*** option if you create the file). Provide the list for each daemon on its command line using the ***-nodirprefs*** option. The distiller daemon is not a multi-threaded application.

Multiple daemons on different hosts

Daemons can monitor separate directories or common directories if the common directory is accessible from each host. When two or more distributed daemons are monitoring the same directory, the distillation parameters each daemon uses to distill files in that directory can differ if distillation parameter overrides are specified on the command line for the daemon, or if the ***-newdprefs*** option is specified. In this case, the resulting distilled files may differ depending on which daemon happened to distill the file.

It may be helpful to first configure the environment for each host by running a daemon in ***-configonly*** mode, specifying all the parameters you want daemons on that host to use, and saving them in a user preferences file. Then invoke the daemon with no command line parameters specified.

It is recommended that all the daemons monitoring a common watched directory have the same settings.

Managing related files

Configuration and log files

When you run multiple daemons, each daemon creates its own configuration and log files. If more than one daemon is configured to use the same directory (or directories) to store the log and configuration files, you must ensure that the file names for each daemon log are unique. One way to do this is to include the ***%P*** specifier in the filename. There are also other considerations for naming and locating these files depending on whether the daemons are all running on the same or host on different hosts.

Multiple daemons on different hosts can use the same directory (or directories) for log and configuration files, as long as the directory is mounted on each host. If you do this, use both the ***%H*** and the ***%P*** specifiers in the log and configuration filename in the *Distill.upr* file, so the various files can be identified by which daemon is using them.

Alternatively, you can safely specify a common name and directory for the log and configuration files for the daemon on each host if the directory is local to each machine, and not a directory that is remotely mounted and shared by more than one of the hosts, for example, */tmp*.

The names and locations of the log and configuration files are defined in the *Distill.upr* file that is read by the daemon during initialization. All daemons started by a given user use the same PostScript startup files. If a single user wants each daemon to store their log and configuration files in a different directory, they can change the filename and directory specifications in the startup files before invoking each new daemon. Multiple users can also invoke their own daemons. They can create the required PostScript startup files (*$HOME/psres/*.upr*) in their home directory and specify the filename and the directory for their log and configuration files.

DTIME.TXT files

When multiple daemons process files in the same watched directory, each daemon overwrites the same *DTIME.TXT* file. If each daemon uses different parameters, you cannot tell which parameters were used on a particular file in that directory. If you must have multiple daemons operating in the same directory, the only way for the parameter information in the *DTIME.TXT* file to be meaningful is to ensure that all the daemons run on the same platform with the same settings.

Written and designed at Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA, 95110-2704, USA

Adobe Systems Europe Limited, Adobe House, Edinburgh EH11 4DU, Scotland, United Kingdom

Adobe Systems Co., Ltd., Yebisu Garden Place Tower, 4-20-3 Ebisu, Shibuya-ku, Tokyo 150, Japan

Printed in the USA.