Adobe PDF Creation Settings

5/7/21 Adobe, Inc.

Contents

List of Examples

1 Introduction

The Adobe PDF Creation Settings file format provides a means of storing settings that affect the creation of PDF files. It is based on the settings (`.joboptions`) files that have been used in all versions of Acrobat Distiller. The format has been expanded so that other applications can share the files and store private settings.

Terminology

Adobe PDF Creation Settings files have an extension of `.joboptions` and have often been called *job options* files (referring to the Distiller *job* that converts PostScript to PDF). Creative Suite applications refer to the files as *presets*. This document refers to the files as *settings files*.

The individual PDF creation settings themselves are represented in the settings files as *key-value pairs*. The key is a name and the value can be any data type supported by PostScript or PDF. In previous versions of *Acrobat Distiller Parameters*, the settings are referred to as *parameters*. This document refers to them as *settings*.

Organization of settings files

A PDF creation settings file contains a number of *settings*. (In the context of Acrobat Distiller, these settings are also referred to as *parameters*.) Each setting has a name, called its *key*, and a *value*, which is data of a defined type. All data types conform to PostScript and PDF syntax, as described in the *PostScript Language Reference* and the *PDF Reference*. All applications in the Adobe Creative Suite, version 2.0 and later, as well as Acrobat Distiller, can read and write settings files.

The settings file is a text file, represented in a format that is recognized by a PostScript interpreter. At the highest level, the file consists of two *dictionaries* that contain key-value pairs representing the settings. The first dictionary is followed by the `setdistillerparams` operator and the second is followed by the `setpagedevice` operator.

```
<<
     key-value pairs representing settings
>> setdistillerparams

<<
     key-value pairs representing settings
>> setpagedevice
```

`setdistillerparams` is a Distiller-specific PostScript operator; that is, it may not be recognized by standard PostScript interpreters. The dictionary that is an operand to `setdistillerparams` contains most of the key-value pairs representing the settings described in this document. For example:

```
<<
     /ASCII85EncodePages false
     /AllowTransparency false
     /AutoRotatePages /None
     /Binding /Left
     ...
```

```
>> setdistillerparams
```

`setpagedevice` is a PostScript operator that allows specifying page device parameters (see the *PostScript Language Reference* for more information). A small number of settings (`HWResolution` and `PageSize`) are specified in the `setpagedevice` dictionary. Unless otherwise specified, this document refers to settings in the `setdistillerparams` dictionary.

Note: Acrobat Distiller has the ability to process `setdistillerparams` operators that appear in PostScript files that it is processing. For example, image compression settings can be changed for specific images. See "How Distiller processes PostScript files" on page 24 for more information.

Namespaces

Prior to Distiller 7.0 and Creative Suite 2.0, all parameters were specified at the root level of the `setdistillerparams` dictionary and were publicly available to Distiller and other applications that used the file.

The concept of *namespaces* has been introduced to allow settings that are private to one or more applications. Each namespace is conceptually separate, which allows multiple applications to define keys with the same name but different meanings.

A single namespace is represented as an entry whose key is `Namespace` and whose value is an array of three required strings that represent the following items, in this order:

- The *creator* of the namespace
- The *name* of the namespace
- The *version number* of the namespace

The following example represents the Common namespace, version 1.0.

```
/Namespace [ (Adobe) (Common) (1.0) ]
```

The version number makes it possible for settings to be added to a namespace or for the values of existing settings to change. In the future, a file might also contain the following code, which would have additional entries not part of the original set.

```
/Namespace [ (Adobe) (Common) (2.0) ]
```

Applications not supporting version 2.0 would ignore them, while newer applications could use them.

Note: The specification allows more than three elements to be stored in the array. It would be possible to refine namespaces further, as in the following example.

```
/Namespace [ (Adobe) (CreativeSuite) (2.0) (Professional) ]
```

Common namespace

All settings supported by Distiller 7.0 and earlier, and documented in past editions of *Acrobat Distiller Parameters*, are considered to be in the `Common` namespace.

- Some settings are applicable to Distiller only (for example, because they make sense only in the context of conversion from PostScript to PDF).
- Other settings have more general applicability, and are supported by one or more of the Creative Suite applications (or may be supported in the future), as indicated in the reference sections of this document.

- Historically, some settings have been provided for specialized use by certain customers and/or OEMs. They do not appear in the user interface nor in preset files provided by Adobe and are expected to be used in highly customized and controlled environments. They are not intended for general use.

The `Namespace` entry appears in the same dictionary as all the other settings defined for that namespace. Therefore, an entry for the `Common` namespace may appear in the top-level dictionary in a settings file, as in the following example:

```
<<
     /ASCII85EncodePages false
     /AllowTransparency false
     /AutoRotatePages /None
     /Binding /Left
     /Namespace [ (Adobe) (Common) (1.0) ]
     ...
>> setdistillerparams
```

In the top-level dictionary, the `Namespace` setting is optional. All settings are assumed to be part of the `Common` namespace. The `Namespace` key is required in the dictionaries in the `OtherNamespaces` array, discussed below.

Other namespaces

The `OtherNamespaces` entry is used to define namespaces other than the `Common` namespace. Its value is an array consisting of one or more dictionaries representing namespaces. Each of the dictionaries must contain the `Namespace` key that identifies the namespace, as well as the settings that are part of the namespace.

Three namespaces have been defined to support Creative Suite 2.0:

`CreativeSuite` (version 2.0): Contains settings that are used by one or more of the suite applications and are not recognized by Distiller.

`InDesign` (version 4.0): contains settings supported by InDesign alone.

The `OtherNamespaces` entry can also be used by third parties and OEMs to place their dictionary of private settings.

Here is an example of the `OtherNamespaces` entry. It contains two namespaces: `InDesign` and `CreativeSuite`:

```
/OtherNamespaces [
  <<
     /AsReaderSpreads false
     /CropImagesToFrames true
     /ErrorControl /WarnAndContinue
     /FlattenerIgnoreSpreadOverrides false
     /IncludeGuidesGrids false
     /IncludeNonPrinting false
     /IncludeSlug false
     /Namespace [
        (Adobe)
        (InDesign)
        (4.0)
     ]
     /OmitPlacedBitmaps false
```

```
      /OmitPlacedEPS false
      /OmitPlacedPDF false
      /SimulateOverprint /Legacy
   >>
   <<
      /AddBleedMarks false
      /AddColorBars false
      /AddCropMarks false
      /AddPageInfo false
      /AddRegMarks false
      /ConvertColors /ConvertToCMYK
      /DestinationProfileName ()
      /DestinationProfileSelector /DocumentCMYK
      /Downsample16BitImages true
      /FlattenerPreset <<
      /PresetSelector /MediumResolution
      >>
      /FormElements false
      /GenerateStructure false
      /IncludeBookmarks false
      /IncludeHyperlinks false
      /IncludeInteractive false
      /IncludeLayers false
      /IncludeProfiles false
      /MultimediaHandling /UseObjectSettings
      /Namespace [
         (Adobe)
         (CreativeSuite)
         (2.0)
      ]
```

Applications that do not recognize the `OtherNamespaces` entry will ignore its contents but should preserve the contents when saving the settings file (as they should preserve all settings); see "Reading and writing settings files" on page 22.

Predefined settings files

Adobe provides a set of predefined PDF settings files or *presets* that you can use for common scenarios. By means of the user interface (UI) of Acrobat Distiller and the Creative Suite applications, you can create and modify new PDF settings files based on the presets. (You should not actually modify the presets themselves.)

The reference chapters, "Common PDF Settings" on page 47 and "Other Namespaces" on page 135, indicate the UI controls, if any, that correspond to each setting. See the Help for these products for detailed information about working with settings files in the user interface.

You can also edit settings files in a text editor. This gives you greater control, because some settings cannot be modified in the UI. However, it also increases the chances of having a settings file that contains syntactic or semantic errors. "How applications handle incorrect settings files" on page 23 explains how Adobe applications deal with errors.

Where presets are installed

When a user installs Distiller or the Creative Suite applications, a group of presets are installed on the user's system. These presets can be shared between applications. "System preset information" on page 20 describes in detail how applications share information about presets.

Each application displays certain presets by default in its user interface. These are stored in a Settings directory, which is typically at these locations:

Windows: C:\Documents and Settings\All Users\Documents\Adobe PDF\Settings\

Mac OS: /Library/Application Support/Adobe PDF/Settings/

Note: The paths specified here represent typical installations. They may differ for specific languages or versions of the products.

Another group of presets is installed in an Extras directory. These can be loaded as needed and typically appear in these locations:

Windows: C:\Documents and Settings\All Users\Documents\Adobe PDF\Extras\

Mac OS: /Library/Application Support/Adobe PDF/Extras/

The table "Common settings files" on page 18 lists presets that are installed in the Common directory. The table "Extra settings files" on page 20 lists presets that are installed in the Extras directory.

In addition to its file name, each preset file has a set of *display names*, one for each language, that are displayed in the application user interface. The first column of the following tables shows the preset file name (minus the `.joboptions` extension), as well as the English display name if different.

Specific applications need not display all presets in their user interface. The third column of each table lists the applications for which the file is visible. For details on how applications show or hide presets, see "System preset information" on page 20).

Common settings files

File name/ English display name	Description	Visible to
High Quality Printing	Creates PDF documents for quality printing on desktop printers and proofers. Created PDF documents can be opened with Acrobat and Adobe Reader 5.0 and later.	All
Smallest File Size	Creates PDF documents best suited for on-screen display, e-mail, and the Internet. Created PDF documents can be opened with Acrobat and Adobe Reader 6.0 and later.	All
Oversized Pages	Creates PDF documents suitable for reliable viewing and printing of engineering drawings larger than 200 x 200 inches. Created PDF documents can be opened with Acrobat and Adobe Reader 7.0 and later.	Distiller
Press Quality	Creates PDF documents best suited for high quality prepress printing. Created PDF documents can be opened with Acrobat and Adobe Reader 5.0 and later.	All

Common settings files

File name/ English display name	Description	Visible to
PDFX1a 2001 PDF/X-1a:2001	Creates PDF documents that are to be checked or must conform to PDF/X-1a:2001, an ISO standard for graphic content exchange. Created PDF documents can be opened with Acrobat or Adobe Reader 4.0 and later.	All
PDFX3 2002 PDF/X-3:2002	Creates PDF documents that are to be checked or must conform to PDF/X-3:2002, an ISO standard for graphic content exchange. Created PDF documents can be opened with Acrobat or Adobe Reader 4.0 and later.	All
Standard	Creates PDF documents suitable for reliable viewing and printing of business documents. Created PDF documents can be opened with Acrobat and Adobe Reader 6.0 and later. With a few exceptions (see "Settings descriptions" on page 47), these settings match the default values for all the settings.	Distiller
PDFA1b 2005 CMYK PDF/A-1b:2005 (CMYK)	Creates PDF documents that are to be checked or must conform to PDF/A-1b, an ISO standard for the long-term preservation (archival) of electronic documents. For more information on creating PDF/A compliant PDF documents, please refer to the Acrobat User Guide. Created PDF documents can be opened with Acrobat and Adobe Reader 5.0 and later.	Distiller
PDFA1b 2005 RGB PDF/A-1b:2005 (RGB)	Same description as PDFA1b 2005 CMYK. See notes following this table.	Distiller

The `PDFA1b 2005 CMYK.joboptions` and `PDFA1b 2005 RGB.joboptions` files are identical except for the `/ColorConversionStrategy` and `/PDFXOutputIntentProfile` entries. The `PDFA1b 2005 CMYK.joboptions` file contains lines,

```
/ColorConversionStrategy /CMYK
/PDFXOutputIntentProfile (U.S. Web Coated (SWOP) v2)
```

while `PDFA1b 2005 RGB.joboptions` contains the lines,

```
/ColorConversionStrategy /sRGB
/PDFXOutputIntentProfile (sRGB IEC61966-2.1)
```

Extra settings files		
File name/ English display name	**Description**	**Visible to**
PDFX1a 2003 PDF/X-1a:2003	Creates PDF documents that are to be checked or must conform to PDF/X-1a:2003, an ISO standard for graphic content exchange. For more information on creating PDF/X-1a compliant PDF documents, please refer to the Acrobat Help. Created PDF documents can be opened with Acrobat or Adobe Reader 5.0 and later.	All
PDFX3 2003 PDF/X-3:2003	Creates PDF documents that are to be checked or must conform to PDF/X-3:2003, an ISO standard for graphic content exchange. For more information on creating PDF/X- 3 compliant PDF documents, please refer to the Acrobat Help. Created PDF documents can be opened with Acrobat or Adobe Reader 5.0 and later.	All
PDFX1a 2001 JPN PDF/X-1a:2001 (Japan)	Japanese version of `PDFX1a 2001`	All
PDFX1a 2003 JPN PDF/X-1a:2003 (Japan)	Japanese version of `PDFX1a 2003`	All
PDFX3 2002 JPN PDF/X-3:2002 (Japan)	Japanese version of `PDFX3 2002`	All
PDFX3 2003 JPN PDF/X-3:2003 (Japan)	Japanese version of `PDFX3 2003`	All
Rich Content PDF	Creates accessible Adobe PDF documents that include tags, hyperlinks, bookmarks, interactive elements, and layers. Created PDF documents can be opened with Acrobat and Adobe Reader 7.0 and later.	All except Illustrator
MAGAZINE Ad 2006 JPN MAGAZINE Ad 2006 (Japan)	Creates PDF documents customized for Japanese Magazine Advertisement. The settings are defined by Japan Magazine Publisher Association Digitizing Promotional Council. Created PDF documents can be opened with Acrobat and Adobe Reader 4.0 and later.	All

System preset information

To facilitate the sharing of settings files, Adobe applications use a mechanism, described in this section, to keep track of the following information:

- The path where the common Settings folder is located.

  Note: The Extras folder is in a location parallel to the Settings folder.

- The display name (the name that should appear in the user interface) that corresponds to each settings (`.joboptions`) file in the Settings directory. For each language for which the application is localized, there is a display name.
- Visibility information about each settings file (that is, whether it should appear in the application's user interface). Settings files that are hidden with respect to an application are said to be *cloaked*. There is visibility information for each application.

This information is equivalent for Windows and Mac OS but is stored in different ways:

- In Windows, the information is in the system registry.
- In Mac OS, the information is in a preference file.

On installation or initialization, Adobe applications instantiate this information if it does not exist. Applications should never assume hard-coded paths but use system calls to obtain the information.

The following sections contain system-specific details.

Windows

In Windows, the top-level registry entry is this setting:

```
My Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\PDF Settings
```

It contains these key items:

- `(Default)`: The full UTF-16 path of the Adobe PDF Settings folder
- A set of key entries for each language (localization). These are `CHS`, `CHT`, `DAN`, `DEU`, `ENU`, `ESO`, `FRA`, `ITA`, `JPN`, `KOR`, `NLD`, `NOR`, `PTB`, `SUO`, and `SVE`. For each of these, the `(Default)` entry is ignored, and the rest of these entries map the name of a settings file (minus the `.joboptions` extension) to the display name for that localization.
- Optionally a `ShowHide` entry that specifies, for each application, which settings files are hidden (cloaked).

The display names (as with all registry strings) are stored as UTF-16 strings. They may be represented as ASCII or in hexadecimal where ASCII is not suitable. The following example shows the entry for Japanese:

```
[HKEY_LOCAL_MACHINE\SOFTWARE\Adobe\PDF Settings\JPN]
     "Standard"=hex:19,6a,96,6e
     "High Quality Print"=hex:d8,9a,c1,54,ea,8c,70,53,37,52
     "Press Quality"=hex:d7,30,ec,30,b9,30,c1,54,ea,8c
     "Smallest File Size"=hex:00,67,0f,5c,d5,30,a1,30,
          a4,30,eb,30,b5,30,a4,30,ba,30
     "PDFX1a 2001"="PDF/X-1a:2001"
     "PDFX3 2002"="PDF/X-3:2002"
     "PDFA1b 2005 CMYK"="PDF/A-1b:2005 (CMYK)"
```

Mac OS

In Mac OS, the preferences file is `com.adobe.AdobePDFSettings.plist`. It is typically stored in `/Library/Preferences/`. (Developers should use the Mac OS APIs to locate this directory rather than assuming a hard-coded path.)

This file contains a dictionary with at least the following keys:

- `AdobePDFSettingsPath`: The full UTF-8 path of the Adobe PDF Settings folder.

- `AdobePDFSettings`: A dictionary containing entries for each localization, where the key is one of the language codes specified above, and the value is an array consisting of pairs of strings, where the first is the file name (less `.joboptions`) and the second is the display name.
- Optionally a `ShowHide` key that specifies, for each application, which settings files are hidden (cloaked).

The following example shows two settings files, `Standard` and `High Quality`, that are localized for English and Spanish.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN"
     "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
     <key>AdobePDFSettings</key>
     <dict>
          <key>ENU</key>
          <array>
               <string>Standard</string>
               <string>Standard</string>
               <string>High Quality</string>
               <string>High Quality Print</string>
          </array>
          <key>ESO</key>
          <array>
               <string>Standard</string>
               <string>Estandar</string>
               <string>High Quality</string>
               <string>Impresion de alta calidad</string>
          </array>
     </dict>
     key>AdobePDFSettingsPath</key>
     <string>/Library/Application Support/Adobe PDF/Settings/</string>
</dict>
</plist>
```

Reading and writing settings files

This section describes the behavior that Adobe applications follow when reading and writing settings files. Many of the principles described are extensions of behavior that has been employed by Distiller since its earliest use of `.joboptions` files.

Compatibility strategies

Applications should provide default values for all settings that are meaningful to them and not present in the settings file. When rewriting a settings file, applications should preserve all settings:

- Settings that are not meaningful to the application should be written out unchanged, along with meaningful settings that have not been changed by the user.
- Settings changed by the user should have their new values saved to the file, assuming the value is of the correct type and is supported by the application.

In some cases, settings have been deprecated and replaced with new settings. Applications should recognize the deprecated versions, if possible, and use them for processing if the functionality is still available. When re-writing files containing such settings, the new variant should be written in addition to, or as a replacement for, the old setting. For an example, see "Using the compliance checking settings" on page 43.

In some cases, different applications may use different settings for similar functionality. For example, Distiller uses settings in the `Common` namespace to perform color conversions. The Creative Suite applications have a larger set of options for color conversion, provided by settings in the `CreativeSuite` namespace. For maximum compatibility with Distiller, the Creative Suite applications write out the most closely matching Common settings when saving settings files. See "Using the color conversion settings" on page 34 for details.

How applications handle incorrect settings files

A settings (`.joboptions`) file can be incorrect in two primary ways:

- The file contains invalid PostScript syntax. PDF creation is never allowed in this case.
- The file contains invalid types or values.

Distiller treats files containing invalid types or values in one of three ways, depending on the settings that are invalid:

- PDF creation is not allowed in cases for which a workaround is not reasonable, given that the resulting PDF file would contain unexpected content.
- A default value is used when the type of a setting is correct but whose incorrect value can be reasonably or predictably converted to another value that does not affect the intent. For example: an application that encounters `/GrayImageFilter /LZWEncode` will substitute `/GrayImageFilter /FlateEncode`.
- All values are converted to defaults when there are more seriously incorrect settings, but not so severe as to require that the PDF file not be created. Many incorrect settings cause Distiller to take this course of action.

The following types of errors are defined for the Creative Suite:

- Syntactic: The settings file contains syntactic errors (that is, invalid PostScript). A warning is generated, and PDF cannot be created using this settings file.
- Catastrophic: Specific settings have values such that Creative Suite applications cannot allow the settings files to be used. The following setting/value combinations result in catastrophic errors:
  - `CompatibilityLevel` is 1.2. Creative Suite only supports 1.3 or greater.
  - `CheckCompliance` is `[(PDF/A-1b:2005 (CMYK))]`. Creative Suite does not support PDF/A.
- Override: The application will use a particular value for a setting regardless of the value in the settings file. For example, Illustrator, InDesign and Photoshop always embed fonts regardless of the value of `EmbedAllFonts`. A warning lets the user know this is occurring.
- Ignore: These do not generate warnings. They involve two categories of settings:
  - The setting is irrelevant to the Creative Suite application, possibly because it involves PostScript processing.
  - The application behaves as though the setting always has a particular value, regardless of its actual value in the settings file. For example, Creative Suite applications always treat the value of `ConvertImagesToIndexed` as if it were `true`.

How Distiller uses Adobe PDF settings

Acrobat Distiller differs from the Creative Suite applications in that it creates PDF files by processing PostScript files. This section provides some information that is specific to the operation of Distiller.

Note: Distiller also provides an API by which programmers can automate its operation. For details on automation methods, see the Acrobat Distiller API Reference.

Distiller initialization

Distiller initialization is controlled by the file `distinit.ps`, which is executed once at Distiller startup. As part of the initialization process, all of the files in the Distiller `Startup` folder are executed. `Example.ps` is one such file.

You can add as many other startup files as you want inside the `Startup` folder. Those files are executed by `distinit.ps` during initialization.

Note: The files in the `Startup` folder are not executed in any specific order—not alphabetically, nor in any other predictable order. For that reason, it is best to add all extra initialization code to the `Example.ps` file to ensure that it all runs in order.

How Distiller processes PostScript files

Distiller may process one or more PostScript files during a single job:

- The first file processed is always the Adobe PDF settings file. The `setdistillerparams` dictionary specifies the initial values for a number of settings. (See "Organization of settings files" on page 14 for details.) For any setting that is not present in the settings file, Distiller assigns its internal *default value* to that setting. "Common PDF Settings" on page 47," lists the default values for all settings.
- The prologue file (`prologue.ps`) is processed if present and the value of the `UsePrologue` setting is `true` in the settings file.
- The main PostScript file is processed.
- The epilogue file (`epilogue.ps`) is processed if present and the value of the `UsePrologue` setting is `true` in the settings file.

Distiller contains a PostScript 3 interpreter with two Distiller-specific operators:

- **`setdistillerparams`**: Attempts to set one or more Distiller parameters (settings). Its operand is a dictionary with one or more key-value pairs, for example:

  ```
  <</CompressPages true>> setdistillerparams
  ```

  Where the value of a key is another dictionary, provide the keys you want to set in that dictionary. For example:

  ```
  <</AutoFilterGrayImages true /GrayACSImageDict <<
  /QFactor 0.25 /HSamples [1 1 1 1] /VSamples [1 1 1 1]>>
  >> setdistillerparams
  ```

- **`currentdistillerparams`**: Returns a dictionary containing key-value pairs for all Distiller parameters (settings). Each execution of this operator allocates and returns a new dictionary.

  Where the value of a key is another dictionary, `currentdistillerparams` returns the key-value pairs in that dictionary.

To enable PostScript files containing the `currentdistillerparams` or `setdistillerparams` operators to be used on PostScript devices such as printers that do not implement these operators, you must add the following definitions to the beginning of the file:

```
/currentdistillerparams where {pop}
{userdict /currentdistillerparams {1 dict} put} ifelse
/setdistillerparams where {pop}
{userdict /setdistillerparams {pop} put} ifelse
```

This PostScript code sequence uses the existing `currentdistillerparams` and `setdistillerparams`, if present. If not, it defines `currentdistillerparams` to return an empty one-element dictionary, and `setdistillerparams` to be a NULL operation.

PostScript language programs that use these operators must not assume that any particular key is present in the dictionary returned by `currentdistillerparams`, or that `setdistillerparams` has any particular side effects.

Modifying settings during the job

The prologue, PostScript, and epilogue files may themselves contain `setdistillerparams` operators that override the initial values of the settings as specified in the settings file (or through the default values). The parameters remain in effect for the duration of the current save level. (See Section 3.7.3 in the *PostScript Language Reference* for a discussion of the save and restore operators.) At the end of the current job, Distiller restores the values to those present before the job began.

Settings specified in the settings file cannot be modified if the `LockDistillerParams` setting is `true`. In addition, a number of settings cannot be modified during a job using `setdistillerparams`. Their values are always taken from the settings file or the default values:

Settings that cannot be modified during a job

AlwaysEmbe	JPEG2000ColorACSImageDict	PDFX1aCheck
AutoRotatePages	JPEG2000ColorImageDict	PDFX3Check
Binding	JPEG2000GrayACSImageDict	PDFXCompliantPDFOnly
CheckCompliance	JPEG2000GrayImageDict	PDFXNoTrimBoxError
ColorConversionStrategy	LockDistillerParams	PDFXTrimBoxtoMediaBoxOffset
CompatibilityLevel	MaxSubsetPc	PDFXSetBleedBoxToMediaBox
CreateJobTicket	NeverEmbed	PDFXBleedBoxToTrimBoxOffset
Description	Optimize	PDFXOutputIntentProfile
DoThumbnails	ParseDSCComments	PDFXTrapped
EmbedAllFonts	ParseDSCCommentsForDocInfo	SubsetFonts

Three of the settings in this list (`DoThumbnails`, `Optimize`, and/or `CompressObjects`) are not used until the post-processing step of distillation.

The remaining Distiller settings can be changed during a job using `setdistillerparams`. One exception is that `PreserveFlatness` cannot be changed in the middle of a page. Its value must be set before any marks are created on the page. For example, when the following PostScript command

sequence is distilled, the value in the `.joboptions` file will be honored and the changes indicated by `setdistillerparams` will not be used:

```
1 1 moveto 1 2 lineto
<< /PreserveFlatness false>> setdistillerparams
20 setflat
```

Any attempt to change `PreserveFlatness` after any marks are drawn on a page is ignored.

Using Distiller to combine PostScript files

Note: Adobe recommends that the Acrobat 7.0 feature Create PDF from Multiple Files be used to combine PostScript and/or PDF files together into one PDF.

Distiller can combine two or more PostScript files to produce a single Adobe PDF document. If the PostScript files have embedded font subsets, Distiller gives the resulting PDF file only one subset for each font. This is much more efficient than creating a set of several PDF documents with duplicate font subsets.

The Acrobat installation provides two files you can modify to combine two or more PostScript files to create a single PDF:

- `Runfilex.ps` combines a set of PostScript files from one or more locations into one PDF file.
- `Rundirex.txt` combines a set of PostScript files from a specific folder or directory into one PDF file.

These files should be located in these directories:

For Acrobat 7 in Windows 2000: \Documents and Settings\All Users\Documents\Adobe PDF 7.0\Example Files

For Acrobat 7 in Windows XP: \Documents and Settings\All Users\Shared Documents\Adobe PDF 7.0\Example Files

In Mac OS: /Library/Application Support/Adobe PDF/Example Files

Follow the instructions in the sample files. The PostScript files are combined in the order in which they are listed. Edit the files in a text editor or word processor, and save the modified files under a new name, as text, with a `.ps` suffix.

You can then use Distiller to convert the combined file to PDF. You can also place the file in an In folder to be converted later.

The conversion settings used are those listed in the Default Settings pop-up menu in the Acrobat Distiller dialog box.

2 Using PDF Creation Settings

This chapter provides information on using PDF creation settings that supplements the information in the settings reference chapters, "Common PDF Settings" on page 47 and "Other Namespaces" on page 135.

Using the image settings

PDF settings files provide several options for the processing of images:

- Images (as well as text and line art) can be *compressed*, thereby significantly reducing the size of a PDF file with little or no loss of detail and precision, depending on the settings chosen.
- Images can be resampled (downsampled or subsampled), which also allows reduction in image size.
- Distiller provides some options relating to bit depth (number of bits per sample) and cropping of images.

The image settings can be specified for these types of images:

Color images: Images that have more than one color component

Grayscale images: Images that have only one color component and more than one bit per sample

Monochrome images: Images that have only one color component and only one bit per sample

The names of the settings indicate which type of image they apply to (for example, `ColorImageFilter`, `GrayImageFilter`, and `MonoImageFilter`).

The image settings are described in detail in "Image settings" on page 58.

Image compression settings

The compression settings fall into several categories. This section describes the overall logic of image compression. There are several options for compression, including JPEG, JPEG2000, CCITTFax, RunLength, Flate, as well as automatic compression. See the following sections for details on each compression type.

The following Boolean settings determine whether compression takes place on the specified image type. If the value is `false`, no compression takes place, and the other compression settings for that image type are ignored:

- `EncodeColorImages`
- `EncodeGrayImages`
- `EncodeMonoImages`

Note: In this document, the term *encode* is used to refer to compression. Strictly speaking, encoding in PDF does not always involve compression.

These settings are Boolean values that determine whether *automatic compression*, in which the producer application chooses compression settings based on the image contents, is applied:

- `AutoFilterColorImages`
- `AutoFilterGrayImages`

See "Automatic compression" on page 19 for details. (Automatic compression is not used for monochrome images.)

When automatic compression is chosen, these settings provide further information:

- `ColorImageAutoFilterStrategy`
- `GrayImageAutoFilterStrategy`
- `ColorACSImageDict`
- `GrayACSImageDict`
- `JPEG2000ColorACSImageDict`
- `JPEG2000GrayACSImageDict`

When automatic compression is not chosen, these settings determine the type of compression (JPEG, Flate, etc.) to be used for the specified image type:

- `ColorImageFilter`
- `GrayImageFilter`
- `MonoImageFilter`

These settings provide further information during non-automatic compression:

- `ColorImageDict`
- `GrayImageDict`
- `JPEG2000ColorImageDict`
- `JPEG2000GrayImageDict`

Images can be compressed using any one of several compression filters. See Section 3.13 of the *PostScript Language Reference* and Section 3.3 of the *PDF Reference* for information on the compression filters.

Note: Because the values of settings can be modified in a PostScript file (see "Modifying settings during the job" on page 21), it is possible when using Distiller to explicitly apply different image settings to specific images. This capability does not apply to Creative Suite applications.

The following sections summarize the types of compression and how Adobe PDF settings can be used to control them.

Flate

Flate (also called Zip) is a compression method that works well on images with large areas of single colors or repeating patterns, such as screen shots and simple images created with paint programs, and for black-and-white images that contain repeating patterns. The Flate method is *lossless*, which means it does not remove data to reduce file size and so does not affect an image's quality.

Adobe's implementation of the Flate filter is derived from the zlib package of Jean-Loup Gailly and Mark Adler.

JPEG

The JPEG compression method is suitable for grayscale or color images, such as continuous-tone photographs that contain more detail than can be reproduced on screen or in print. JPEG is a *lossy* compression method that can achieve much smaller file sizes than Flate compression, which is lossless. JPEG attempts to reduce file size with the minimum loss of information.

JPEG encoding and decoding is done by means of the direct cosine transformation (DCT) algorithm. This algorithm can take several optional parameters. In PostScript files, these parameters are contained in the `DCTEncode` parameter dictionary that is used by the `DCTEncode` filter. See "DCTEncode Filter" in Section 3.13.3 of the *PostScript Language Reference* for detailed information.

Four PDF settings are dictionaries that specify parameters to control JPEG compression. They are `ColorACSImageDict` and `GrayACSImageDict` (for automatic compression) and `ColorImageDict` and `GrayImageDict` (for non-automatic compression). These dictionaries are based on the `DCTEncode` parameter dictionary.

The default value for each of these dictionaries is

```
<</Qfactor 0.76 /Hsamples [2 1 1 2] /Vsamples [2 1 1 2]>>
```

The following should be noted about these dictionaries:

- The `QFactor` entry is the only one that can be set directly. It provides a measure of the trade-off between image compression and image quality. Lower values of `QFactor` mean higher quality and therefore less compression.
- `HSamples` and `VSamples` can be set in the PDF settings file. However, Distiller and other applications ignore these values and provide their own values based on `QFactor`. If `QFactor` >= 0.5, both the `HSamples` and `VSamples` arrays are set to [2 1 1 2]. If `QFactor` < 0.5, then both the `HSamples` and `VSamples` arrays are set to [1 1 1 1]. If you save the settings to a file, the computed values for `HSamples` and `VSamples` are saved in the file, regardless of the original values present in the file.
- The other entries that can appear in a `DCTEncode` parameter dictionary are not settable through these image dictionaries. They include `Columns`, `Rows`, `Colors`, `QuantTables`, `HuffTables`, `ColorTransform`, and `CloseTarget`. These parameters are set internally in Distiller (or other application) depending on the properties of each image. `ColorTransform` is set to the "best" value for each image. It is set to 0 if the color space is Lab or Gray or (CMYK AND QFactor >= 0.5). Otherwise, `ColorTransform` is set to 1.

In the user interface of Distiller and the Creative Suite applications, you can use the `Quality` field to achieve one of five levels of image quality. The following table shows the values of `HSamples`, `VSamples`, and `QFactor` that correspond to Minimum, Low, Medium, High, and Maximum image quality.

Image compression quality

Quality	HSamples	VSamples	QFactor
Minimum	[2 1 1 2]	[2 1 1 2]	2.40
Low	[2 1 1 2]	[2 1 1 2]	1.30
Medium	[2 1 1 2]	[2 1 1 2]	0.76
High	[1 1 1 1]	[1 1 1 1]	0.40
Maximum	[1 1 1 1]	[1 1 1 1]	0.15

Note: When Distiller processes PostScript files to produce PDF, it normally decompresses all JPEG images and then recompresses them according to the settings in effect. The exception is when the `PassThroughJPEGImages` setting is `true`. Illustrator and InDesign do not use this setting but normally behave as if it were `true` with regard to placed PDF files containing compressed images. That is, they do not uncompress and recompress them unless color conversion or downsampling takes place. See the reference entry for `PassThroughJPEGImages` for more information.

JPEG2000

JPEG2000 is a new international standard for the compression and packaging of image data. It defines a wavelet-based method for image compression that gives somewhat better size reduction than other methods such as JPEG or CCITT. It is suitable both for images that have a single color component and for those with multiple color components. JPEG2000 is especially well suited for color images with smooth variation in color values.

There is no filter name defined for JPEG2000 compression in the PostScript language definition). PDF files use the `JPXDecode` filter to decompress JPEG2000 images. See the *PDF Reference* for information about JPEG2000 compression in PDF files. See also http://www.jpeg.org/JPEG2000.htm.

The JPEG2000 compression filter provides the ability to encode different versions of an image with varying resolution. For example, a thumbnail version of the image may be encoded in the data, followed by a sequence of other versions of the image, each with approximately four times as many samples (twice the width, twice the height) as the previous one. The last version is the highest resolution image, corresponding to the value of the `Quality` key (see the following table). A PDF viewer may not need to decode the highest resolution version but only the resolution that best matches the current viewing or printing needs. Therefore, fewer bytes need to be processed, a particular benefit when viewing files over the Web. JPEG2000 data also has a built-in tiling structure, such that if the full image is not visible, only those tiles being displayed need to be decoded (to an appropriate resolution).

There are four PDF settings that specify dictionaries for customizing color or grayscale image compression for the JPEG2000 filter:

- `JPEG2000ColorImageDict` and `JPEG2000GrayImageDict` are used with regular (non-automatic) compression.
- `JPEG2000ColorACSImageDict` and `JPEG2000GrayACSImageDict` are used with automatic compression.

These dictionaries have three entries you can set, as shown in the following table. Since all entries are optional, an empty dictionary is acceptable.

Entries in JPEG2000 image dictionaries

Key	Type	Value
`TileWidth`	integer	*(Optional)* The width of JPEG2000 image tiles in samples. Valid values are 128 - 2048; values outside this range generate a range error. Default value: 256.
`TileHeight`	integer	*(Optional)* The height of JPEG2000 image tiles in samples. Valid values are 128 - 2048; values outside this range generate a range error. Default value: 256.
`Quality`	integer	*(Optional)* The required image quality for the highest resolution image in the image progression. Valid values are 1 - 100, where 1 is the lowest quality (highest compression), 99 means visually lossless compression, and 100 means numerically lossless compression. Default value: 15 (Medium). In the Compression panel of the Distiller UI, the mapping that occurs for the predefined options is as follows: • Minimum = 5 • Low = 10 • Medium = 15 • High = 20 • Maximum = 30 • Lossless = 100

The user interface provides a Tile Size option if `CompatibilityLevel` is set to 1.5 or higher and the Compression setting is JPEG2000 or Automatic (JPEG2000). The amount specified sets both the `TileWidth` and `TileHeight` parameters to the same value. If a settings file has been modified so that the values are different, Distiller accepts both values, but Creative Suite applications use the value of `TileWidth` for both.

Automatic compression

Automatic compression for color or grayscale bitmap images means that the application producing the PDF determines the compression filters to be applied to individual images. Setting `AutoFilterColorImages` and/or `AutoFilterGrayImages` to `true` causes automatic compression to take place for color and grayscale images, respectively.

You can use the `ColorImageAutoFilterStrategy` and `GrayImageAutoFilterStrategy` settings to choose between two automatic compression strategies. The value of these settings can be either `JPEG` (the default) or `JPEG2000` (which applies only to PDF 1.5 and later files). If you choose `JPEG`:

- JPEG compression (the `DCTEncode` filter) is used for 8-bit images that have smooth color changes (low-frequency images). The parameters specified in the `ColorACSImageDict` or `GrayACSImageDict` dictionary are used to provide further control. JPEG typically provides greater compression than Flate, but is *lossy* (can lose information).
- Flate compression is used for all other images. Flate is a lossless compression method, so it is more suitable for images with sharp color changes (high-frequency images). Flate does not take any additional parameters.

  Note: Flate compression is also used when the image uses a `DeviceN` color space, is small (< 1024 bytes), extremely wide (> 40000 pixels) or is ChromaKeyed.

If you choose `JPEG2000`:

- Lossy JPEG2000 compression is used for low-frequency images. The compression parameters specified in the `JPEG2000ColorACSImageDict` or `JPEG2000GrayACSImageDict` dictionary provide further control of the compression. See "JPEG2000" on page 17 for information about these dictionaries.
- Lossless JPEG2000 compression is used for high-frequency images. The compression `JPEG2000ColorACSImageDict` or `JPEG2000GrayACSImageDict` dictionaries are used as well, with the modification that the `Quality` parameter is forced to 100 (to achieve lossless compression).

Non-automatic compression

This section describes the compression options that are available when automatic compression is not chosen. (Automatic compression does not apply to monochrome images.)

Color and grayscale images

Grayscale images have one color component and more than one 1 bit per component. Color images have more than one color component and 1 or more bits per component:

- For grayscale images that have 2 or 4 bits per component or color images with 1, 2, or 4 bits per component, only Flate compression is permitted
- For grayscale or color images with 8 bits per component, JPEG, JPEG2000, and Flate are permitted

  Note: For grayscale or color images with more than 8 bits per component, the least significant bits of each image sample are removed, yielding 8 bits per sample.

When image compression is selected (with `EncodeColorImages`, `EncodeGrayImages`, or `EncodeMonoImages`), the `ColorImageFilter`, `GrayImageFilter`, or `MonoImageFilter` settings specify which compression filter should be used. If no filter name is specified (is absent), lossless Flate is used in all cases. Invalid filter names generate an error.

Note: The following filters are *never* selected, even if they are specified in the Adobe PDF settings file: `LZWEncode`, `ASCII85Encode`, and `ASCIIHexEncode`.

Under the following conditions, `FlateEncode` is used even if another filter is specified:

- The selected filter is `CCITTFaxEncode` and the image is wide (more than 40,000 columns).
- The selected filter is `JPXEncode` and the image is indexed or ChromaKeyed or `CompatibilityLevel` is less than 1.5.
- The selected filter is `DCTEncode` and the image is wide (more than 40,000 columns), indexed, deviceN or ChromaKeyed.
- The selected filter is not supported for the number of colors or sample depth of the particular image being compressed.

Monochrome (black and white) images

Monochrome images are defined as images with only one color component and one bit per sample. For monochrome image compression, the available filters are `CCITTFaxEncode`, `RunLengthEncode`, and `FlateEncode`.

The `CCITTFaxEncode` parameter dictionary specifies options for CCITT compression. See "CCITTFaxEncode Filter" in Section 3.13.3 of the *PostScript Language Reference* for details. The `MonoImageDict` setting is a dictionary that contains the same keys as the `CCITTFaxEncode` parameter dictionary; any of the keys can be customized.

`CCITTFaxEncode` (CCITT Group 4) compression typically yields the best compression of monochrome images. It is specified by a value of -1 for the `K` key in the `CCITTFaxEncode` parameter dictionary, for two-dimensional encoding. A value of 0 for this key corresponds to CCITT Group 3 (one-dimensional encoding).

Note: With the exceptions of the `AntiAliasMonoImages` and `MonoImageDepth` parameters, the monochrome image compression parameters also can be applied to stencil masks created by the `imagemask` operator. Parameter behavior is the same in both cases. For details on `imagemask`, see the *PostScript Language Reference*.

Downsampling and subsampling images

Downsampling and subsampling are processes that reduce the number of pixels per inch in an image. To do so, pixels in a sample area are combined to make one larger pixel.

The following subsampling and downsampling methods are available:

Subsampling: A pixel in the center of the sample area replaces the entire area at the specified resolution. Subsampling is significantly faster than downsampling but results in images that are less smooth and continuous.

Average downsampling: The pixels in a sample area are averaged, and the average pixel color replaces the entire area at the specified resolution.

Bicubic downsampling: A weighted average is used to determine pixel color and usually yields better results than the simple averaging method of downsampling. This is the slowest but most precise method, resulting in the smoothest tonal gradations.

You should downsample or subsample bitmap images when they are sampled at a higher resolution than the output device supports. The excess data increases the time it takes the device to process the image

without improving image quality. For example, by reducing an image from a typical printer resolution of 300 pixels per inch to a typical monitor resolution of 72 pixels per inch, the amount of data needed to represent an image is decreased by a factor of 16, and the image can be drawn on the screen much more quickly.

Downsampling settings

These settings are Boolean values that specify whether images of the specified type should be downsampled: `DownsampleColorImages`, `DownsampleGrayImages`, and `DownsampleMonoImages`.

These settings specify the resolution to which images should be downsampled: `ColorImageResolution`, `GrayImageResolution`, or `MonoImageResolution`.

These settings specify the type of sampling (average or bicubic downsampling, subsampling, or none) `ColorImageDownsampleType`, `GrayImageDownsampleType`, or `MonoImageDownsampleType`.

In order for downsampling to actually occur, the ratio of the input image resolution to the desired output resolution (specified by the above parameters) must exceed the *downsampling threshold*. These settings are used to set the downsampling threshold resolution: `ColorImageDownsampleThreshold`, `GrayImageDownsampleThreshold`, and `MonoImageDownsampleThreshold`.

For example, if `ColorImageResolution` is 72 and `ColorImageDownsampleThreshold` is set to 1.5, an image is not downsampled unless its input resolution is greater than 108 pixels per inch:

```
trunc((72 * 1.5) + .5) = 108 pixels per inch
```

Threshold values must be between 1.0 through 10.0, inclusive, with a default value of 1.5. (If you set the threshold out of range, it reverts to 1.5.)

Controlling the range of bit depths for which downsampling occurs

You can also control the range of bit depths for which downsampling occurs. For example, in a workflow where there is a mixture of 1-bit and 8-bit data, you can downsample the 8-bit data while not touching the 1-bit data. This is done with the following settings:

- `ColorImageMinDownsampleDepth` can be 1, 2, 4, or 8
- `GrayImageMinDownsampleDepth` can be 2, 4, or 8

For example, a value of 4 for `ColorImageMinDownsampleDepth` means that only 4- and 8 bits-per-sample color images are downsampled (assuming `DownsampleColorImages` is `true`). Similarly, a value of 8 for `GrayImageMinDownsampleDepth` means that only 8 bits-per-sample gray images are downsampled (assuming `DownsampleGrayImages` is `true`).

Note: 12 bits-per-sample images (valid in PostScript) are treated exactly as 8 bits-per-sample images because they are converted to 8 bits per sample before downsampling takes place.

Setting compression of text, line art, and objects

You can use the `CompressPages` setting to set the compression of text and line art. For PDF 1.5 and above, you can use the `CompressObjects` setting to control object-level compression, which is the consolidation of small objects that cannot be individually compressed into *streams* that can then be efficiently compressed.

Distiller-only image settings

The following options apply only to Distiller and are not supported by Creative Suite applications.

Controlling bit depth

Bit depth is the number of bits used to represent each color component of each sample of an image. (For example, red, green, and blue are the color components in an RGB image). Distiller supports the control of bit depth by means of the `ColorImageDepth`, `GrayImageDepth`, and `MonoImageDepth` settings.

The bit depth of an image can be decreased (for example, from 8 bits per sample to 4 bits per sample) to save space, regardless of whether the image is downsampled.

If an image is downsampled, the bit depth can be increased to provide *anti-aliasing*. Anti-aliasing increases the number of bits per sample to preserve some of the information that is otherwise lost by downsampling. Anti-aliasing occurs only in the following conditions:

- The image depth setting specifies a bit depth greater than that of the incoming image.
- The value of the appropriate setting `AntiAliasColorImages`, `AntiAliasGrayImages`, or `AntiAliasMonoImages` is `true`. (They need not be `true` to decrease the bit depth.)
- Sampling is enabled and the downsample thresholds are met; therefore, sampling actually occurs.

In these cases, Distiller first increases the bit depth, then downsamples the image.

For example, suppose a 300 pixel-per-inch monochrome image is downsampled to 150 pixels per inch. If `MonoImageDepth` is 4 and `AntiAliasMonoImages` is `true`, the bit depth of the image is increased prior to downsampling so that it becomes a 4-bit grayscale image. Each of the samples in the downsampled image is produced from four samples in the input image; because each of the input samples can be either on or off, there are 16 possible values for each sample in the downsampled image.

Note that after the bit depth settings have been applied, an input grayscale or monochrome image may be changed to the other type.

- A grayscale image specified to have a `GrayImageDepth` of 1 is treated as a monochrome image.
- A monochrome image that has a `MonoImageDepth` of 2, 4, or 8 becomes a grayscale image.

Distiller determines whether to apply compression settings *after* downsampling has taken place. If the bit depth has changed, the resulting image type determines which encoding setting is examined. For example, if a monochrome image was changed to have a bit depth of 2 or more, the `EncodeGrayImages` setting would be checked. If encoding is enabled, the image is compressed using the filter type and filter parameter dictionary specified by the settings for the resulting image type.

The PostScript example below shows a code fragment specifying that monochrome images be downsampled to 72 pixels per inch, converted to 2 bits per sample, and encoded using Flate compression. Because the downsampled images are grayscale, the filter is specified using the grayscale rather than the monochrome image settings. Also, assuming that the input image is a 300-pixels-per-inch image, it is downsampled to 75 pixels per inch, the closest possible value to the 72 pixels per inch requested.

```
<<    /DownsampleMonoImages true
      /MonoImageResolution 72
      /MonoImageDepth 2
      /EncodeGrayImages true
      /AntiAliasMonoImages true
```

```
        /GrayImageFilter /FlateEncode
>> setdistillerparams
```

Specifying a minimum resolution of sampled images

In addition to the downsampling settings, starting with version 7.0, Distiller provides settings to check whether images meet a minimum resolution.

`ColorImageMinResolution`, `GrayImageMinResolution`, and `MonoImageMinResolution` are settings that specify an integer between 9 to 64000 representing the minimum resolution for an image.

`ColorImageMinResolutionPolicy`, `GrayImageMinResolutionPolicy`, and `MonoImageMinResolutionPolicy` are settings that specify what happens when images are found that do not meet the minimum resolution. They are names that can take one of the following values:

`OK`: the minimum resolution settings are ignored.

`Warning`: Any image with a resolution smaller than the specified minimum generates a warning when the PDF file is created.

`Error`: Any image with a resolution smaller than the specified minimum generates an error, and the job fails with a limit check error.

The default values for these settings in the predefined Adobe PDF settings files are chosen to be the same as the values for the default downsampling resolution. With these default values, Distiller's default behavior does not change; that is, Distiller does not enforce any lower limit on image resolution.

These settings can be used to ensure that a PDF file does not have any images with lower resolution than the defined limit. This feature is primarily for prepress people who want to detect that no low resolution images make it into a PDF file. An example is a sampled image in an advertisement where the image must be of a certain quality.

If you get a warning or error about a low resolution image and the settings are correct according to your requirements, you need to go to the source of the image and regenerate it with a higher resolution. Distiller cannot actively alter these images because it doesn't support up-sampling.

In the following example, Distiller issues a warning every time a sampled gray image with resolution smaller than 100 ppi is placed in the PDF file:

```
/GrayImageMinResolution 100
/GrayImageMinResolutionPolicy /Warning
```

The warning messages will look like this:

```
%%[ Warning: Gray image resolution (92 ppi) is
lower than /GrayImageMinResolution (100 ppi) ]%%
```

If `GrayImageMinResolutionPolicy` is set to `Error`, then an error message is emitted and the job fails with a `limitcheck` error. If `GrayImageMinResolutionPolicy` is set to `OK`, then distillation continues normally.

Note: While there are no dependencies or interaction between the downsampling settings and the minimum resolution settings, normally you would not set the resolution policy to `Warning` or `Error` and at the same time set the minimum resolution to a value that is higher than the downsampling threshold. If you do this, the result is that *all* images are flagged as having too low a resolution. If the resolution policy is `Error`, then only PDF files with no images would be distilled.

Controlling downsampling and encoding for each sampled image

You can separately control the downsampling and encoding of each sampled image in a PostScript file. To do this, you must make adjustments to the Distiller parameters in the file just before, and appropriate to, each image.

Disabling of image cropping

Distiller determines whether more than 10 percent of an image sample falls outside the existing clip path. If so, Distiller normally discards (crops) the image samples that fall outside the clip area, resulting in smaller images and PDF files.

For workflows in which the full-size (non-cropped) images must be extracted for special-purpose image manipulation, it is possible to disable cropping, using the settings `CropColorImages`, `CropGrayImages`, and `CropMonoImages` for color, grayscale, and monochrome images, respectively. These settings are Boolean values:

`false`: indicates that Distiller should not clip image samples regardless of the current clip area.

`true`: (the default) indicates that Distiller should crop only if the 10 percent criteria is met.

Note: InDesign uses a separate setting, `CropImagesToFrames`, to control cropping.

Using the font settings

Fonts can be included (*embedded*) in a PDF file to ensure that the file can be rendered correctly, regardless of whether the fonts are installed on the machine used to view the file. For example, the exact font may be needed to achieve certain effects such as high-end printing or to ensure portability in situations where the viewer cannot create a substitute font.

Distiller supports the `EmbedAllFonts` setting, which specifies whether fonts should be embedded. Other Creative Suite applications always embed fonts when possible.

Note: Embedding is subject to license; specific fonts can indicate that embedding is not permitted.

See "Font settings" on page 93 for a description of each of the font settings.

Embedded fonts make a PDF file larger. To produce files as small as possible, fonts can be *subsetted*. When you subset a font, only the information required to draw glyphs (specific renderings) for the characters used in the document is embedded. Subsetting is expressed as a percentage of the font glyphs for a font format. The `SubsetFonts` and `MaxSubsetPct` settings are used to control partial embedding of fonts.

Distiller supports additional settings to control which fonts are embedded. The rest of this section describes how Distiller chooses whether to embed fonts.

Distiller maintains lists of fonts that will be embedded or not embedded. `AlwaysEmbed` specifies fonts that should always be embedded, and `NeverEmbed` specifies fonts that should never be embedded. These two settings are arrays that contain a list of font names. Optionally, the first element in the arrays may be a Boolean value (`true` or `false`).

- If the first element is not a Boolean value, the array of font names represents the entire list of fonts to be embedded or not embedded.
- If the first element is the Boolean `true`, the font names in the array are *added to* Distiller's internal list of fonts to be embedded (`AlwaysEmbed`) or not embedded (`NeverEmbed`).

- If the first element is the Boolean `false`, the font names in the array are *removed from* Distiller's internal list of fonts to be embedded (`AlwaysEmbed`) or not embedded (`NeverEmbed`).

If a font appears in both the `NeverEmbed` and `AlwaysEmbed` lists, it is never embedded.

The `EmbedAllFonts` setting is a Boolean value that, when `true`, specifies that all fonts be embedded except those in the `NeverEmbed` array.

Note: A font may not be embedded if its license doesn't permit embedding, even though its name is in the `AlwaysEmbed` list or `EmbedAllFonts` is `true`. Furthermore, a symbolic font is always embedded (if license permits) even if its name is in the `NeverEmbed` list.

In this PostScript example, Minion Regular is always embedded, and ITC Stone Serif Italic and ITC Stone Sans are never embedded.

```
<< /AlwaysEmbed [ /Minion-Regular ]
     /NeverEmbed [ /StoneSans /StoneSerif-Italic ]
>> setdistillerparams
```

Note: The font name given to definefont does not have to match the name in the FontInfo dictionary. For instance, in this example the full name of the font defined as 'StoneSans' is 'ITC Stone Sans.'

The following table identifies the types of fonts that you can (or cannot) embed or subset through Distiller settings.

Distiller control over embedding and subsetting fonts

Font	NeverEmbed?	AlwaysEmbed?	Subset?
Type 1	Yes	Yes	Yes
Type 3	No - Always embedded		No - Always subsetted
True Type (Type 42)	Yes	Yes	No - Always subsetted
CIDFontType0	Yes	Yes	No - Always subsetted
CIDFontType1	No - Always embedded		No - Always subsetted
CIDFontType2	Yes	Yes	No - Always subsetted
OpenType	Yes	Yes	Yes

For additional information on Type 1, Type 3, Type 42, and CID-keyed fonts, see Chapter 5, "Fonts," in the *PostScript Language Reference* and Chapter 5 in the *PDF Reference*. You also can find additional documentation on fonts at the Acrobat Developer Center.

Note: Distiller 5 and above also support OpenType fonts; Distiller 4 does not. OpenType fonts are based on the compact font format (CFF). For more information, see the *Compact Font Format Specification* at the Acrobat Developer Center.

Using the color conversion settings

This section describes how the color conversion settings are used and explains the correspondence between different groups of settings.

Acrobat Distiller uses a number of settings to control color conversion. As with all Distiller parameters, these settings are defined in the `Common` namespace (see "Common PDF Settings" on page 47" for details).

Creative Suite applications have a sophisticated user interface for determining color conversions when producing PDF files. The options provided in the UI correspond to settings in the `CreativeSuite` namespace of the settings files (see "Other Namespaces" on page 135" for details). These options provide a superset of the functionality provided by the Distiller settings in the `Common` namespace. To maximize interoperability, Creative Suite applications store approximations to their color settings in the Common settings when saving settings files.

Distiller color conversion settings

Distiller uses a number of settings related to color. The following settings are only used by Distiller: `DefaultRenderingIntent`, `ParseICCProfilesInComments`, `PreserveDICMYKValues`, `PreserveHalftoneInfo`, `TransferFunctionInfo`, and `UCRandBGInfo`. They control such features as whether Distiller preserves (that is, passes into the PDF file) halftoning, overprinting, and transfer function information. See "Common PDF Settings" on page 47 for information on these settings.

Other color conversion settings are shared to a limited degree with Creative Suite applications. They are:

`ColorSettingsFile`: A file containing color settings. When a color settings file is specified, all other color conversion settings are ignored and not selectable in the UI. The Creative Suite applications recognize the existence of this setting as an indication that the user has modified color settings outside the suite.

`CalCMYKProfile`, **`CalGrayProfile`**, **`CalRGBProfile`**: Settings that specify the names of ICC profiles to be used for tagging or converting CMYK, gray, or RGB color data, respectively.

`sRGBProfile`: The name of an ICC profile to use for converting color spaces to CalRGB (PDF 1.2) or sRGB (PDF 1.3 and above).

Note: The Creative Suite applications do not support saving files as PDF 1.2.

`ColorConversionStrategy`: Specifies a strategy for determining output color family and color space and the inclusion of ICC profiles. (See "ICCBased Color Spaces" in Section 4.5.4 of the *PDF Reference* for details on profiles.) The `ColorConversionStrategy` setting has the following possible values.

Value	UI equivalent
`LeaveColorUnchanged`	Leave Color Unchanged
`UseDeviceIndependentColor`	Tag Everything for Color Management (no conversion)
`UseDeviceIndependentColor-ForImages`	Tag Only Images for Color Management (no conversion)
`sRGB`	Convert All Colors to sRGB
`CMYK`	Convert All Colors to CMYK

Note: Distiller leaves `Separation` and `DeviceN` color spaces unchanged in PDF output. Creative Suite applications convert the alternate color spaces; for example, when converting to CMYK, the alternate color space is changed to CMYK if necessary and the tint transform is adjusted accordingly.

The following table shows how Distiller converts the PostScript input to the equivalent color space for each `ColorConversionStrategy` value. The notes below the table provide further information.

PS color space (in) vs. PDF color space (out)

PS Input	LeaveColor Unchanged	UseDevice Independent Color	UseDevice Independent ColorForImages	sRGB	CMYK
Gray text and graphics	DeviceGray	ICCBased	DeviceGray	DeviceGray	DeviceGray
Gray image	DeviceGray	ICCBased	ICCBased	DeviceGray	DeviceGray
RGB text and graphics	DeviceRGB	ICCBased	DeviceRGB	sRGB	DeviceCMYK
RGB image	DeviceRGB	ICCBased	ICCBased	sRGB	DeviceCMYK
CMYK text and graphics	DeviceCMYK	ICCBased	DeviceCMYK	sRGB	DeviceCMYK
CMYK image	DeviceCMYK	ICCBased	ICCBased	sRGB	DeviceCMYK
CIE text and graphics	ICCBased	ICCBased	ICCBased	sRGB	DeviceGray/ DeviceCMYK[a]
CIE image	ICCBased	ICCBased	ICCBased	sRGB	DeviceGray/ DeviceCMYK[b]

a. CIEBasedA becomes DeviceGray; others become DeviceCMYK.
b. CIEBasedA becomes DeviceGray; others become DeviceCMYK.

Notes on the PS color space (in) vs. PDF color space (out) table:

- ICCBased color spaces were introduced in PDF 1.3. When creating PDF 1.2 files using device-independent colors, the color spaces CalGray (for gray), CalRGB (for RGB), or Lab (for CMYK) are used in place of ICCBased.
- *sRGB* is an industry standard color space, but PDF does not have a color space by this name. Instead, it can be represented precisely in PDF as an ICCBased color space or approximated by a CalRGB color space. During conversion, Distiller chooses either CalRGB or ICCBased as appropriate. (For PDF 1.2, it must choose CalRGB.)

Creative Suite color conversion settings

Creative Suite applications use several color conversion settings. The settings, which are in the `CreativeSuite` namespace, determine whether colors should be converted and which profiles should be included for which objects. They appear in the Output panel of the PDF export dialog box in the user interface. This section explains how the settings are used.

Converting colors

The `ConvertColors` setting determines whether colors should be converted. A value of `NoConversion` corresponds to "No Color Conversion" in the UI. Values of `ConvertToCMYK` and `ConvertToRGB` can correspond to either of the following UI settings, as follows:

Convert to Destination: All colors are converted to destination profile space unless profiles are same as destination profile. (Native and untagged placed objects are treated as if tagged with the corresponding document profile.)

Convert to Destination (Preserve Numbers): (This option is not used by Photoshop.) Colors are converted to the destination profile if the color space family (for example, CMYK) does not match the destination color space family. Colors are not converted if there is no embedded profile or if the object is native (that is, created in the application itself as opposed to placed graphics such as images or PDF).

Including profiles

This section describes the settings that control whether and which color profiles should be included.

`IncludeProfiles` is a Boolean value. If it is `false`, no profiles are included in the generated PDF. The UI setting is "Don't Include Profiles". If `IncludeProfiles` is `true` and colors are being converted to a destination, the UI specifies "Include Destination Profiles."

If `IncludeProfiles` is `true` and colors are not being converted, the options are:

Include All Profiles: Includes profiles for all content.

Include Tagged Source Profiles: Leaves device-dependent colors unchanged and preserves device-independent colors as the nearest possible equivalent in PDF.

Include All RGB and Tagged Source CMYK Profiles: Includes profiles for tagged RGB and tagged CMYK objects, as well as the Document RGB profile for untagged RGB objects.

These correspond to additional settings, explained below.

Note: In the Photoshop UI, only "Include Destination Profile" is available when converting colors.

`UntaggedRGBHandling` and `UntaggedCMYKHandling` determine what should happen to untagged RGB or CMYK objects during conversion. They can either be left untagged (`LeaveUntagged`) or tagged with the document profile (`UseDocumentProfile`).

The choice of profiles is controlled by `DestinationProfileSelector`, which can take these values:

- `NA` means that no color conversion takes place (`ConvertColors` is `NoConversion`).
- `WorkingCMYK`, `WorkingRGB`, `DocumentCMYK`, and `DocumentRGB` specify the profile to be used for color conversion. When using `WorkingCMYK` or `DocumentCMYK`, Creative Suite applications also write the profile name to `DestinationProfileName`.
- `UseName` means that the ICC profile specified by `DestinationProfileName` should be used for color conversion.

Color settings interchange

This section describes how Creative Suite applications decide whether to use the `CreativeSuite` namespace settings or the `Common` settings and what values they store when saving settings files.

When common settings are used

Creative Suite applications use the Common settings when the CreativeSuite settings are not present in the settings file or when there is an inconsistency between the two types of settings. This section describes when the Common settings are used.

The following settings indicate that the settings file was not created by a Creative Suite application:

- A value for `ColorSettingsFile` other than the empty string or `(None)`. In this case, the other Common settings are used.
- A value of `UseDeviceIndependentColorForImages` for `ColorConversionStrategy`. In this case, Creative Suite applications override this value and use `LeaveColorsUnchanged`.

In other cases, the Common settings and CreativeSuite settings are present but inconsistent, indicating that the settings file must have been modified subsequent to being written by a Creative Suite application. In these cases, `ColorSettingsFile` is empty or unspecified and the following values are present:

- The value of `ColorConversionStrategy` is `UseDeviceIndependentColorForImages`. In this case, Creative Suite applications override this value and use `LeaveColorsUnchanged`.
- The value of `ColorConversionStrategy` is `CMYK` and the value of `ConvertColors` (Creative Suite) is `ConvertToRGB` or `NoConversion`.
- The value of `ColorConversionStrategy` is `sRGB` and the value of `ConvertColors` is `ConvertToCMYK` or `NoConversion`.
- The value of `ColorConversionStrategy` is `LeaveColorsUnchanged` and the value of `ConvertColors` is `ConvertToCMYK` or `ConvertToRGB`.
- The value of `ColorConversionStrategy` is `LeaveColorsUnchanged`, the value of `ConvertColors` is `NoConversion`, `IncludeProfiles` is `true`, and `UntaggedCMYKHandling` is `UseDocumentProfile`.
- The value of `ColorConversionStrategy` is `UseDeviceIndependentColor` and the value of `ConvertColors` is `ConvertToCMYK` or `ConvertToRGB`.
- The value of `ColorConversionStrategy` is `UseDeviceIndependentColor`, the value of `ConvertColors` is `NoConversion`, and `IncludeProfiles` is `false`.
- `CalCMYKProfile` has a non-empty value, `DestinationProfileName` has a non-empty value that does not match `CalCMYKProfile` , and the value of `ConvertColors` is `ConvertToCMYK`.

In all other cases, the CreativeSuite settings are used.

When the Common settings are used, the following table shows how the Common setting `ColorConversionStrategy` determines the values of the Creative Suite UI elements. (See the table "Conversion from CreativeSuite settings to Common settings" on page 30 for additional information.) Note that in all cases, if color management is off, the Profile Inclusion Policy defaults to "Don't Include Profiles."

Creative Suite equivalents for ColorConversionStrategy

ColorConversionStrategy	UI elements
`LeaveColorsUnchanged`	Color Conversion = No Conversion Profile Inclusion Policy = Include Tagged Source Profiles
`UseDeviceIndependentColor`	Color Conversion = No Conversion Profile Inclusion Policy = Include All Profiles
`UseDeviceIndependentColorForImages`	Color Conversion = No Conversion Profile Inclusion Policy = Include Tagged Source Profiles
`sRGB`	Color Conversion = Convert to Destination Profile Inclusion Policy = Include All Profiles Destination Profile = sRGBProfile
`CMYK`	Color Conversion = Convert to Destination (Preserve Numbers) Profile Inclusion Policy = Don't Include Profiles Destination Profile = CalCMYKProfile

Note: When specifying PDF/X-1a compliance (that is, the value of `CheckCompliance` is either [`PDFX1a:2001`] or [`PDFX1a:2003`]), Creative Suite applications always use `CMYK` as the value of `ColorConversionStrategy`. See "Using the standards settings" on page 33 for more information.

Saving common settings equivalents

When saving settings files, the Creative Suite applications write the best possible approximations of their color settings to the Common settings. `ColorSettingsFile` is always set to `()`. This section describes how the other settings are determined.

The following table shows the relationship between the Creative Suite UI, the Creative Suite settings and the Common settings. The first four columns show the possible values of `ConvertColors`, `IncludeProfiles`, `UntaggedCMYKHandling`, and `UntaggedRGBHandling`. The last column shows the Common settings that correspond to them. The first column also shows what UI names correspond to the groups of settings.

Conversion from CreativeSuite settings to Common settings

ConvertColors /UI Name	**Include Profiles**	**UntaggedCMYK Handling**	**UntaggedRGB Handling**	**Common settings**
`NoConversion` Don't Include Profiles	`false`	`LeaveUntagged`	`LeaveUntagged`	`ColorConversionStrategy = LeaveColorsUnchanged`
Include Tagged Source Profiles	`true`	`LeaveUntagged`	`LeaveUntagged`	
Include All RGB & Tagged Source CMYK Profiles	`true`	`LeaveUntagged`	`UseDocumentProfile`	

Include All Profiles	`true`	`UseDocumentProfile`	`UseDocumentProfile`	`ColorConversionStrategy = UseDeviceIndependentColor`
				`CalCMYKProfile = Document CMYK`
				`CalRGBProfile = Document RGB`

`ConvertToCMYK` Convert To Destination	`false`	`UseDocumentProfile`	`UseDocumentProfile`	`ColorConversionStrategy = CMYK` `CalCMYKProfile =` destination CMYK profile name `CalRGBProfile =` document RGB profile name
	`true`	`UseDocumentProfile`		
Convert To Destination (Preserve Numbers)	`false`	`LeaveUntagged`		
	`true`	`LeaveUntagged`		
`ConvertToRGB` Convert To Destination	`false`	`UseDocumentProfile`	`UseDocumentProfile`	`ColorConversionStrategy = RGB` `CalCMYKProfile =` document CMYK profile name `CalRGBProfile =` document RGB profile name
	`true`	`UseDocumentProfile`	`UseDocumentProfile`	
`ConvertToRGB` Convert To Destination (Preserve Numbers)	`false`	`UseDocumentProfile`	`LeaveUntagged`	
	`true`	`UseDocumentProfile`	`LeaveUntagged`	

Using the advanced Adobe PDF settings

You can customize advanced Adobe PDF settings. When the `CreateJobTicket` setting is `true`, Distiller produces *internal job tickets* (that is, job tickets within the PDF file). Job ticket keys are created in response to `setpagedevice` keys and DSC comments.

The relationship between `setpagedevice` keys and job ticket keys, and the relationship between DSC comments and job ticket keys is described in the following sections. For details on the format and contents of job tickets, see *Portable Job Ticket Format, Version 1.1*.

Relationship between setpagedevice keys and job ticket keys

The following table lists the `setpagedevice` keys that Distiller supports and describes where in an internal job ticket Distiller stores the corresponding key values.

Note: `setpagedevice` keys that are distilled into the JobTicketContents dictionary rather than into the PageRange dictionary must appear in the first page of the PostScript job; otherwise, they are ignored. In the `PS page` column of the following table, "First" identifies `setpagedevice` keys that must appear on the first page.

Relationship between setpagedevice keys and job ticket keys

setpagedevice key	PS page	job ticket key
`Bind` **Note:** `Bind` is unrelated to the `Binding` setting.	First	`JobTicketContents::Finishing`
`CutMedia`	First	If the value of the `CutMedia setpagedevice` key is less than 4, Distiller represents the `setpagedevice` value in `JobTicketContents::MediaUsage::CutMedia`. Otherwise, it represents the value in `JobTicketContents::PrintLayout::Signature::Sheets::MediaUsage::CutMedia`.
`DeviceRenderingInfo/ValuesPerColorComponent`	Any	`PageRange::Rendering::ValuesPerColorComponent`
`Duplex`	First	`JobTicketContents::PrintLayout` See Appendix B.4 in the *Portable Job Ticket Format, Version 1.1*, for a description of the general appearance of a job ticket that can produce duplex printing.
`Fold`	First	`JobTicketContents::Finishing`
`HWResolution`	Any	`PageRange::Rendering::Resolution`
`Jog`	First	`JobTicketContents::Finishing`
`Laminate`	First	`JobTicketContents::Finishing`
`ManualFeed`	First	`JobTicketContents::MediaSource::ManualFeed`
`MediaClass`	First	`JobTicketContents::MediaSource::MediaClass`
`MediaColor`	First	`JobTicketContents::MediaSource::MediaColor`

setpagedevice key	PS page	job ticket key
`MediaPosition`	First	`JobTicketContents::MediaSource::Position`
`MediaType`	First	`JobTicketContents::Media::Category`
`MediaWeight`	First	`JobTicketContents::Media::Weight`
`MirrorPrint`	First	`JobTicketContents::MediaUsage::MirrorPrint`
`NegativePrint`	First	`JobTicketContents::MediaUsage::NegativePrint`
`PageSize`	Any	`PageRange::MediaBox`
`PostRenderingEnhance`	Any	`PageRange::Rendering::PostRenderingEnhance`
`PreRenderingEnhance`	Any	`PageRange::Rendering::PreRenderingEnhance`
`ProcessColorModel`	Any	`PageRange::ColorModel::ProcessColorModel`
`SeparationColorNames`	Any	`PageRange::ColorModel::ColorantParams`
`SeparationOrder`	Any	`PageRange::ColorModel::ColorantOrder`
`Separations`	Any	`PageRange::ColorModel::Separations`
`Staple`	First	`JobTicketContents::Finishing`
`Trapping`	Any	`PageRange::Trapping::Trapping`
`TrappingDetails`	Any	`PageRange::Trapping::TrappingDetails`
`Trim`	First	`JobTicketContents::Finishing`
`Tumble`	First	`JobTicketContents::PrintLayout` Such a job ticket is identical to that described for the `Duplex setpagedevice` key, except the `CTM` for the `Back` surface is rotated 180 degrees.

Relationship between PostScript comments and job ticket keys

When the `ParseDSCComments` setting is `true`, Distiller interprets certain PostScript comments to produce true job ticket `PlaneOrder` objects. Such PostScript comments include `%%Page:` (which is more specifically a DSC comment), `%%QRKPageBegin:`, and `%%PlateColor:`. Distiller also supports the `%%PlateColor:` *PostScript comment*; however, use of that comment is discouraged.

Using the standards settings

The Standards settings provide control over PDF/A- and PDF/X-compliant output:

- PDF/A is a proposed ISO standard for the long-term preservation (archival) of electronic documents.
- PDF/X is a focused subset of PDF designed specifically for reliable prepress data interchange. It is an International Standards Organization (ISO) standard (www.iso.org).

See "Standards settings" on page 125 for details on the settings that are relevant to standards compliance.

Using the compliance checking settings

In Acrobat 6, the `PDFX1aCheck` and `PDFX3Check` settings were introduced to check for compliance with PDF/X-1a (2001) and PDF/X-3 (2002), respectively.

The `CheckCompliance` setting was introduced in Distiller 7 and is supported by Creative Suite applications. It specifies the standard against which the document's compliance is checked. It is an array of strings, each of which is the name of a standard.

Note: Currently only one string may appear in the array.

In Distiller 7 and the Creative Suite, `CheckCompliance`, if present, takes precedence over `PDFX1aCheck` and `PDFX3Check`:

- `/CheckCompliance [ /PDFX1a:2001 ]` has the same meaning as `/PDFX1aCheck true`
- `/CheckCompliance [ /PDFX3:2002 ]` has the same meaning as `/PDFX3Check true`
- Other values of `CheckCompliance` have no corresponding Distiller 6 values.

If a settings file contains `CheckCompliance` and not `PDFX1aCheck` or `PDFX3Check`, the appropriate values of `PDFX1aCheck` and `PDFX3Check` are written out when the file is saved to provide backward compatibility with Distiller 6 for testing of PDF/X-1a:2001 and PDF/X-3:2002 standards compliance. That is,

```
/CheckCompliance [ /PDFX1a:2001 ]
```

without `PDFX1aCheck` and `PDFX3Check` is written out and also generates:

```
/PDFX1aCheck true
/PDFX3Check false
```

Similarly:

```
/CheckCompliance [ /PDFX3:2002 ]
```

without `PDFX1aCheck` and `PDFX3Check` is written out and also generates:

```
/PDFX1aCheck false
/PDFX3Check true
```

Any other values for `CheckCompliance` also generates:

```
/PDFX1aCheck false
/PDFX3Check false
```

If `PDFX1aCheck` and/or `PDFX3Check` are present, they are preserved when the file is saved.

If `CheckCompliance` is not present in a file:

- If `PDFX1aCheck` is present and `true`, `CheckCompliance` takes the value `[/PDFX1a:2001]`.
- Otherwise, if `PDFX3Check` is `true`, `CheckCompliance` takes the value `[/PDFX3:2002]`.
- Otherwise, `CheckCompliance` takes the value `[/None]`.

Using the PDF/X output intent settings

In a PDF/X compliant file, the document catalog must contain an `OutputIntents` entry that specifies a *PDF/X output intent dictionary*. Several settings are used to specify the entries in this dictionary. The rest of this section explains how they work.

Distiller uses the following settings to create the output intent dictionary: `PDFXOutputIntentProfile`, `PDFXOutputCondition`, `PDFXOutputConditionIdentifier`, and `PDFXRegistryName`. For compatibility, these settings are also used by Creative Suite applications , along with the Creative Suite-specific `PDFXOutputIntentProfileSelector` setting.

For Distiller only, these settings are ignored in the case where the output intent dictionary is specified explicitly in the PostScript file (by means of `pdfmark` operators). The rest of this discussion assumes the PostScript file does not specify this information.

The PDF/X output intent dictionary is described in Section 10.10.4 of the *PDF Reference*. It specifies the following entries (note that these are PDF names):

`OutputConditionIdentifier`: A string identifying the intended printing condition of the document. Typically, it is the reference name of a standard production condition in an industry-standard registry such as the ICC Characterization Data Registry (see `RegistryName` below). It can be specified by the `PDFXOutputConditionIdentifier` setting.

`DestOutputProfile`: A PDF/X *output intent profile*, which is a stream representing an ICC profile that defines the transformation from the PDF document's source colors to the output device colorants. It is not required if `OutputConditionIdentifier` specifies a standard production condition. However, Creative Suite applications always store the profile.

`OutputCondition`: An optional human-readable comment describing the printing condition. It can be specified by the `PDFXOutputCondition` setting.

`RegistryName`: A string identifying the registry that defines the condition defined by `OutputConditionIdentifier`. It can be specified by the `PDFXRegistryName` setting.

`Info`: A string containing additional information. Distiller and Creative Suite aplications use this entry to store the profile name.

The `PDFXOutputIntentProfile` setting is used to identify a profile name. It may have one of the following values:

- `(None)` or the empty string `()` are supported by Distiller only. This value means that the PostScript document must specify the output intent destination profile for PDF/X validation to succeed.
- `(Use Output Condition Identifier)` is also supported by Distiller only and overridden by Creative Suite applications (as described in the rest of this section). In this case, Distiller uses the value defined by `PDFXOutputConditionIdentifier` and a profile is not included in the PDF file.
- The name of the output intent destination profile. This profile is embedded as the value of the `DestOutputProfile` entry.

  Note: For Distiller only, if the profile corresponding to the name is not present on the system, Distiller stores the profile name in `OutputConditionIdentifier`.

Creative Suite applications use the `PDFXOutputIntentProfileSelector` (in the `CreativeSuite` namespace) to identify the profile, which can be a name specified by `PDFXOutputIntentProfile` or a reference to the Working CMYK, Working RGB, Document CMYK, or DocumentRGB profile.

As mentioned above, Creative Suite applications do not support the value of `(Use Output Condition Identifier)` for `PDFXOutputIntentProfile`. When they encounter it in a settings file, they override it as follows:

- `PDFXOutputIntentProfileSelector` and `DestinationProfileSelector` are set to `UseName`.
- If `PDFXOutputConditionIdentifier` specifies a known condition (that is, one that maps to a specific set of characterization data at www.color.org (the ICC web site), then `PDFXOutputIntentProfile` and `DestinationProfileName` are set to the name of the profile associated with the condition. The following table shows a list of profiles and their corresponding characterization data:

ICC Profile	Characterization data (output condition identifier)
U.S. Web Coated (SWOP) v2	CGATS TR 001
Euroscale Coated v2	FOGRA1
Euroscale Uncoated v2	FOGRA4
Europe ISO Coated FOGRA27	FOGRA27
Japan Color 2001 Coated	JC200103
Japan Color 2001 Uncoated	JC200104
Japan Color 2002 Newspaper	JCN2002

- If `PDFXOutputConditionIdentifier` specifies an unknown condition and `DestinationProfileName` is a "PRTR" profile and maps to an unknown condition, then `PDFXOutputIntentProfile` is set to the value of `DestinationProfileName`.
- If `PDFXOutputConditionIdentifier` specifies an unknown condition and `DestinationProfileName` is not a "PRTR" profile or maps to a known condition, then
  - If `ColorConversionStrategy` is `CMYK`, `PDFXOutputIntentProfile` and `DestinationProfileName` are set to `(U.S. Web Uncoated V2)`.
  - If `ColorConversionStrategy` is `sRGB`, `PDFXOutputIntentProfile` and `DestinationProfileName` are set to `(ROMM RGB)`.

  Note: For Creative Suite applications, when color management is on and PDF/X compliance has been specified, the effective profiles specified by `DestinationProfileName`, `CalCMYKProfile` and `PDFXOutputIntentProfile` must be the same.

In addition, when specifying PDF/X-1a compliance (that is, the value of `CheckCompliance` is either `[PDFX1a:2001]` or `[PDFX1a:2003]`), if the value of `ColorConversionStrategy` is not `CMYK`, the Creative Suite applications use the following values:

- `ColorConversionStrategy` = `CMYK`
- `ConvertColors` = `ConvertToCMYK`
- `UntaggedCMYKHandling` = `LeaveUntagged`
- `UntaggedRGBHandling` = `UseDocumentProfile`
- `IncludeProfiles` = `false`

Distiller examples

The following examples show how Distiller sets the values in the output intent dictionary.

Example: *Setting the output intent dictionary to Euroscale Uncoated v2*

In this example, the `PDFXOutputIntentProfile` is set to `(Euroscale Uncoated v2)`, whose corresponding output condition identifier (`FOGRA4`) is known by Distiller.

```
12 0 obj
<<
     /Type /OutputIntent
     /S /GTS_PDFX
     /OutputConditionIdentifier (FOGRA4)
     /RegistryName (http://www.color.org)
     /Info (Euroscale Uncoated v2)
     /DestOutputProfile 11 0 R
>>
endobj
11 0 obj
<<
     /N 4
     /Length 388226
     /Filter /FlateDecode
>>
stream
... ICCProfile data ...
endstream
endobj
```

Example: *Setting the output intent dictionary to U.S. Web Uncoated v2*

In this example, `PDFXOutputIntentProfile` is set to `(U.S. Web Uncoated v2)` and Distiller does not know the corresponding output condition identifier. In this case, the value of the output condition identifier is set to the profile name.

```
12 0 obj
<<
     /Type /OutputIntent
     /S /GTS_PDFX
     /OutputConditionIdentifier (U.S. Web Uncoated v2)
     /Info (U.S. Web Uncoated v2)
     /DestOutputProfile 11 0 R
>>
endobj
11 0 obj
<<
     /N 4
     /Length 386435
     /Filter /FlateDecode
>>
stream
... ICCProfile data ...
endstream
endobj
```

3 Common PDF Settings

This chapter describes the Adobe PDF settings that belong to the Common namespace. (See "Namespaces" on page 11.) These settings have historically been documented in *Acrobat Distiller Parameters*. They are supported by Distiller and may be supported by one or more of the applications in the Creative Suite, version 2.0 and later. The settings in other namespaces are described in "Other Namespaces" on page 135.

The settings are grouped into the same categories that appear in the Distiller user interface (UI). The Creative Suite applications have similar interfaces for PDF creation settings, although they may differ in some respects. More information on the use of Adobe PDF settings can be found in Help for Distiller and the Creative Suite applications.

Settings descriptions

The possible values for the settings are enumerated, with their meanings.

Supported by

Indicates which of the applications (Distiller, Illustrator, InDesign, and Photoshop) support this setting.

Type

The type of data. The types are consistent with PostScript syntax and can include Boolean, number, string, array, and dictionary.

UI name

The name (if any) of the user interface control that corresponds to this setting. (There can be slight variations based on application)

Default value

Each setting has a default value, which is the value that is used when a setting is not specified in the settings file.

Note: The default values are the same as those in the `Standard.joboptions` file, with the following exceptions:

- `NeverEmbed` defaults to [true] (no list of fonts)
- `Description` is not provided
- `TransferFunctionInfo` defaults to Preserve
- `CompressObjects` defaults to Off
- `CalGrayProfile` defaults to ()
- `PassThroughJPEGImages` defaults to false

General settings

These settings are available in the General panel of the Distiller settings dialog box. Those settings that are supported by Creative Suite applications are indicated.

AutoRotatePages

Allows Distiller to automatically orient (rotate) pages based on the predominant text orientation. Auto-rotation is not done if the file contains the `%%ViewingOrientation` DSC comment and `ParseDSCComments` is true. If `AutoRotatePages` is set to `None`, pages are not automatically oriented and the `%%ViewingOrientation` DSC comment is ignored.

Value	UI equivalent	Description
`None`	Off	No automatic page rotation.
`All`	Collectively by File	All pages are rotated to match the predominant text orientation.
`PageByPage`	Individually	Pages are rotated on a page-by-page basis. This value is useful for mixed portrait and landscape documents.

Supported by

Distiller

Type

name

UI name

Auto-Rotate Pages

Default value

`All`

Binding

Controls the value of the `PageDirection` entry in the `ViewerPreferences` dictionary of the PDF file. `PageDirection` determines how the printed pages would be bound.

Value	UI equivalent	Description
`Left`	Left	For left binding
`Right`	Right	For right binding

Note: InDesign does not use this setting but sets the binding through other UI controls.

Supported by

Distiller

Type

name

UI name

Binding

Default value

`Left`

CompatibilityLevel

The PDF version number: 1.2, 1.3, 1.4, 1.5, 1.6, or 1.7.

Note: Applications other than Distiller do not support saving files as PDF 1.2.

Note: If you create files with Acrobat 7.0 compatibility, the resulting PDF files may not be compatible with earlier Acrobat versions.

Supported by

all applications

Type

real

UI name

Compatibility

Default value

```
1.4
```

CompressObjects

Controls object-level compression, introduced in PDF 1.5. PDF objects that are not individually compressible can be combined into *object streams* that can be efficiently compressed. Only PDF consumers that support PDF 1.5 can process object streams.

Value	UI equivalent	Description
`Off`	Off	PDF 1.5 object compression is not used.
`Tags`	Tags Only	PDF 1.5 object compression is applied to the logical structure tree in a document. Versions 5 and earlier of Acrobat and Adobe Reader can open and use these documents, but cannot use the logical structure (tag) information. Versions 6 and later have full access to the tag information.
`All`	Maximum	Maximum compression of objects is performed. The compressed file is readable by Acrobat and Adobe Reader version 6 and later. Available only if `CompatibilityLevel` is 1.5 or higher.

Note: Creative Suite applications do not provide a user interface for this setting. For InDesign, if `GenerateStructure` (in the `CreativeSuite` namespace) is `true`, Tagged PDF is generated and this setting is set to `Tags` (if `CompatibilityLevel` is 1.5 or higher). A value of `All` will be treated as `Tags` by Creative Suite applications.

Supported by

all applications

Type

name

UI name

Object-Level Compression Distiller

Default value

`Off`

CoreDistVersion

(Read only) Version number of the Distiller implementation. This is neither the version number of the PostScript interpreter used in Distiller nor the version number displayed in the UI.

Supported by

Distiller

Type

integer

Default value

7000

Description

Provides a description of the Adobe PDF settings file that can be displayed in the UI. Each key in the dictionary is a standard 3-letter language code; for example, `ENU` for English (see the Acrobat and PDF Library API Reference for a listing of these codes). The value associated with each language key is a string that contains the description of the settings file in that language. It is assumed that the string will be reflowed to fit the width of the display field.

Supported by

all applications

Type

dictionary

UI name

Description

DoThumbnails

If `true`, thumbnails are created for the pages of the resulting PDF file.

Supported by

all applications

Type

Boolean

UI name

Distiller: Embed Thumbnails
Creative Suite: Embed Page Thumbnails

Default value

`false`

EndPage

The last page of the document to be produced.

`StartPage` and `EndPage` together determine the range of pages in the PostScript file that are converted to PDF. If `StartPage` is greater than 1 (the default), no PDF output is produced for the first (`StartPage -1`) pages of PostScript. `StartPage` becomes page 1 of the PDF file. If `EndPage` is greater than `-1`, distilling stops after the `EndPage` of PostScript. Distiller checks these two settings at the time that the first PostScript marking operator is executed in a job.

Note: `StartPage` and `EndPage` are useful when debugging PostScript. They are not recommended for general purpose use, as Distiller does not retain page number references in document links.

Supported by

Distiller

Type

integer

UI name

All Pages, Pages From: To:

Default value

`-1`

ExportLayers

Controls the layer output of exported spreads, based on their visibility & printability layer attributes.

Supported by

InDesign

Type

Name

UI name

Export Layers

Default value

`ExportVisiblePrintableLayers`

HWResolution

Provides the resolution for the PDF file if this value has not already been supplied by the PostScript file itself.

Note: This setting appears as part of the `setpagedevice` dictionary in a settings file, as described in "Organization of settings files" on page 10. See *PostScript Language Reference* for more information.

Supported by

Distiller

Type

array

UI name

Resolution

Default value

`[600 600]`

ImageMemory

The number of bytes in the buffer used in the sample processing of color, grayscale, and monochrome images. When the buffer is full, Distiller writes its contents to disk.

The value of this setting should be between `0` and `1048576`. If it is set to a negative integer, zero is used.

Supported by

Distiller

Type

integer

Namespace

This setting identifies the namespace to which all other settings in the same settings dictionary belong. This setting may appear in any namespace. For the `Common` namespace, it is optional. For all other namespaces, it is required. See "Namespaces" on page 11 for more information.

Supported by

all applications

Type

array

Default value

```
[ (Adobe) (Common) (1.0) ]
```

Optimize

If `true`, the PDF file is optimized for fast web viewing. This process is also referred to as *linearization* (see the *PDF Reference* for detailed information.

Supported by

all applications

Type

Boolean

UI name

Optimize For Fast Web View

Default value

`true`

OtherNamespaces

This setting is an array containing additional settings dictionaries in namespaces other than the current namespace. See "Namespaces" on page 11 for more information.

Supported by

all applications

Type

array

PageSize

Provides the page size for the PDF file if this value has not already been supplied by the PostScript file itself.

Note: This setting appears as part of the `setpagedevice` dictionary in a settings file, as described in "Organization of settings files" on page 10. See *PostScript Language Reference* for more information.

Supported by

Distiller

Type

array

UI name

Default Page Size

Default value

```
[612.000 729.000]
```

StartPage

See the description of the `EndPage` setting.

Supported by

Distiller

Type

integer

UI name

All Pages, Pages From: To:

Default value

```
1
```

Image settings

This section lists the Adobe PDF settings that apply to each image type. In Distiller, these options appear in the "Images" panel of the user interface. For the Creative Suite applications, these settings appear in the "Compression" panel. (Photoshop does not present different options for the different image types but uses the settings that apply to the active image.)

For more information on how these settings are used together, see "Using the image settings" on page 23.

Color image settings

This section lists the compression and downsampling settings for color images (images that have more than one color component).

AntiAliasColorImages

If `true`, Distiller permits anti-aliasing on color images.

Anti-aliasing increases the number of bits per component in downsampled images to preserve some of the information that is otherwise lost by downsampling. Anti-aliasing is only performed if the image is actually downsampled and `ColorImageDepth` has a value greater than the number of bits per color component in the input image.

For more information on anti-aliasing see "Controlling bit depth" on page 31.

Supported by

Distiller

Type

Boolean

Default value

`false`

AutoFilterColorImages

If `true`, the compression filter for color images is chosen based on the properties of each image, in conjunction with the `ColorImageAutoFilterStrategy` setting. See "Automatic compression" on page 28 for more information.

If `false`, all color sampled images are compressed using the filter specified by `ColorImageFilter`.

This setting is relevant only if `EncodeColorImages` is `true`.

Supported by

all applications

Type

Boolean

UI name

Compression: Automatic (JPEG) & Automatic (JPEG2000)

Default value

`true`

ColorACSImageDict

Dictionary of parameters for JPEG compression when JPEG is chosen from the Automatic filter selection (see `AutoFilterColorImages` on page 26). `ColorACSImageDict` is based on the `DCTEncode` parameter dictionary described in Section 3.13.3 in the *PostScript Language Reference*.

See "Monochrome (black and white) images" on page 29 for details on the use of this dictionary.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality

Default value

```
<</Qfactor 0.76 /Hsamples [2 1 1 2] /Vsamples [2 1 1 2]>>
```

ColorImageAutoFilterStrategy

Specifies the automatic compression strategy that is used when `AutoFilterColorImages` is set to `true`. Must be one of the following values:

Value	UI equivalent	Description
`JPEG`	Automatic (JPEG)	Lossy JPEG compression is used for low-frequency images and lossless Flate compression for high-frequency images.
`JPEG2000`	Automatic (JPEG2000)	Applies only when `CompatibilityLevel` is set to 1.5 or higher. Lossy JPEG2000 compression is used for low-frequency images (images with smooth color changes) and lossless JPEG2000 compression for high-frequency images.

See "Automatic compression" on page 28 for more information.

Supported by

Distiller, Illustrator, InDesign

Type

name.

UI name

Compression

Default value

`JPEG`

ColorImageDepth

Specifies the number of bits per color component in the output image. See "Controlling bit depth" on page 31 for more information.

Allowed values are:

- The number of bits per sample: `1`, `2`, `4`, or `8`.
- `-1`, which forces the downsampled image to have the same number of bits per color component as the original image

Note: Photoshop uses the `Downsample16BitImages` setting to decrease the bit depth of 16-bit images to 8 bits per sample so that JPEG compression can be used.

Supported by

Distiller

Type

integer

Default value

```
-1
```

ColorImageDict

Dictionary of parameters for JPEG compression. `ColorImageDict` is based on the `DCTEncode` parameter dictionary described in Section 3.13.3 in the *PostScript Language Reference*.

See "Monochrome (black and white) images" on page 29 for details on how this dictionary is used.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality

Default value

```
<</Qfactor 0.76 /Hsamples [2 1 1 2] /Vsamples [2 1 1 2]>>
```

ColorImageDownsampleThreshold

Sets the downsample threshold for color images. This is the ratio of image resolution to output resolution above which downsampling may be performed. Must be between 1.0 through 10.0, inclusive. If you set the threshold out of range, it reverts to a default of 1.5.

See "Downsampling and subsampling images" on page 29 for details on using this setting.

Supported by

all applications

Type

number

UI name

pixels per inch; for images above: *value* pixels per inch

Default value

1.5 (UI shows 225 pixels-per-inch)

ColorImageDownsampleType

Must be one of the following values:

Value	UI equivalent	Description
`Average`	Average Downsampling to	Groups of samples are averaged to get the new downsampled value.
`Bicubic`	Bicubic Downsampling to	Bicubic interpolation is used on a group of samples to get a new downsampled value.
`Subsample`	Subsampling to	The center sample from a group of samples is used as the new downsampled value.
`None`	Distiller: Off CS: Do Not Downsample	No sampling

Supported by

all applications

Type

name

UI name

Sampling

Default value

`Bicubic`

ColorImageFilter

Specifies the compression filter to be used for color images. Ignored if `AutoFilterColorImages` is `true` or `EncodeColorImages` is `false`.

Must be one of the following values:

Value	UI equivalent	Description
`DCTEncode`	JPEG	Selects JPEG compression.
`FlateEncode`	ZIP	Selects Flate (ZIP) compression.
`JPXEncode`	JPEG2000	Selects JPEG2000 compression.

Note: JPEG2000 options only appear in UI if `CompatibilityLevel` is set to 1.5 or higher.

`DCTEncode` is used only if the output image has 8 bits per color component; otherwise `FlateEncode` is used.

For compatibility with Distiller 3.0 Adobe PDF settings files, Distiller 6.0 and later (as well as Creative Suite applications) revert to Flate compression if this setting is set to `LZWEncode`. Other values cause Distiller to stop with a range error.

Supported by

all applications

Type

name

UI name

Compression

Default value

`DCTEncode`

ColorImageMinDownsampleDepth

If `DownsampleColorImages` is `true`, controls the range of bit depths for which color image downsampling occurs. Valid values are 1, 2, 4 or 8.

For more information, see "Controlling the range of bit depths for which downsampling occurs" on page 30.

Supported by

Distiller

Type

integer

Default value

`1`

ColorImageMinResolution

Imposes a lower limit to the resolution of sampled images. Valid values are from 9 to 64000, inclusive.

How this value is used is determined by `ColorImageMinResolutionPolicy`. For more information, see "Specifying a minimum resolution of sampled images" on page 32.

Supported by

Distiller

Type

integer

UI name

Policy->Color Image Policy

Default value

`150`

ColorImageMinResolutionPolicy

Sets the policy for imposition of a lower limit to the resolution of sampled color images as specified by the `ColorImageMinResolution` setting. The following table shows the valid names.

Value	UI equivalent	Description
`OK`	Ignore	Distiller's default behavior does not change—i.e., Distiller does not enforce any lower limit on image resolution, ignoring any value specified by `ColorImageMinResolution`.
`Warning`	Warn and continue	A warning is issued every time a sampled color image with resolution smaller than the value specified by `ColorImageMinResolution` is placed in the PDF file. The job continues after issuing the warning.
`Error`	Cancel job	An error occurs when a sampled color image with resolution smaller than the value specified by `ColorImageMinResolution` is placed in the PDF file. The job fails with a limit check error.

For more information, see "Specifying a minimum resolution of sampled images" on page 32.

Supported by

Distiller

Type

name

UI name

Policy->Color Image Policy

Default value

`OK`

ColorImageResolution

Specifies the resolution to which downsampled color images are reduced; valid values are 9 to 2400 pixels per inch. A color image is downsampled if `DownsampleColorImages` is `true` and the resolution of the input image meets the criteria described in "Downsampling and subsampling images" on page 29.

Supported by

all applications

Type

integer

UI name

Sampling: pixels per inch

Default value

`150`

ConvertImagesToIndexed

If `true`, Distiller converts images that use fewer than 257 colors to an indexed color space for compactness. This conversion, when enabled, produces smaller PDF files but may make distillation slower.

Note: Creative Suite applications behave as if this setting is always `true`.

Supported by

Distiller

Type

Boolean

Default value

`true`

CropColorImages

If `CropColorImages` is `false`, color images are never cropped, whether or not the current clip would remove any image samples.

For more information, see "Disabling of image cropping" on page 33.

Supported by

Distiller

Type

Boolean

Default value

`true`

DownsampleColorImages

If `true`, color images are downsampled using the resolution specified by `ColorImageResolution`, assuming the threshold specified by `ColorImageDownsampleThreshold` is met. If `false`, downsampling is not done and the resolution of color images in the PDF file is the same as the source images.

See "Downsampling and subsampling images" on page 29 for more information.

Supported by

all applications

Type

Boolean

UI name

Sampling

Default value

`true`

EncodeColorImages

If `true`, color images are encoded using the compression filter specified by the value of `ColorImageFilter`. If `false`, compression filters are not applied to color images.

Supported by

all applications

Type

Boolean

UI name

Compression

Default value

`true`

JPEG2000ColorACSImageDict

Dictionary of parameters for automatic JPEG2000 compression of color images. See "JPEG2000" on page 26 for details.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality, Tile Size

Default value

```
<</TileWidth 256/TileHeight 256/Quality 15 >>
```

JPEG2000ColorImageDict

Dictionary of parameters for JPEG2000 compression of color images. See "JPEG2000" on page 26 for details.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality, Tile Size

Default value

```
<</TileWidth 256/TileHeight 256/Quality 15 >>
```

Grayscale image settings

This section lists compression and downsampling settings for grayscale images (images with only one color component and more than one bit per sample).

AntiAliasGrayImages

If `true`, Distiller permits anti-aliasing on grayscale images. Anti-aliasing increases the number of bits per component in downsampled images to preserve some of the information that is otherwise lost by downsampling. Anti-aliasing is only performed if the image is actually downsampled and `GrayImageDepth` has a value greater than the number of bits per sample in the input image. For more information, see "Controlling bit depth" on page 31.

Supported by

Distiller

Type

Boolean

Default value

`false`

AutoFilterGrayImages

If `true`, the compression filter for gray images is chosen based on the properties of each image, in conjunction with the `GrayImageAutoFilterStrategy` setting. See "Automatic compression" on page 28 for more information. If `false`, all color sampled images are compressed using the filter specified by `GrayImageFilter`. This setting is relevant only if `EncodeGrayImages` is `true`.

Supported by

all applications

Type

Boolean

UI name

Compression

Default value

`true`

CropGrayImages

If `false`, gray images are never cropped, whether or not the current clip would remove any image samples. See "Disabling of image cropping" on page 33 for more information.

Supported by

Distiller

Type

Boolean

Default value

`true`

DownsampleGrayImages

If `true`, grayscale images are downsampled using the resolution specified by `GrayImageResolution`. If `false`, downsampling does not occur, and the resolution of grayscale images in the PDF file is the same as the source images.

Supported by

all applications

Type

Boolean

UI name

Sampling

Default value

`true`

EncodeGrayImages

If `true`, grayscale images are encoded using the compression filter specified by the value of `GrayImageFilter`. If `false`, compression filters are not applied to grayscale sampled images.

Supported by

all applications

Type

Boolean

UI name

Compression

GrayACSImageDict

Dictionary of parameters for JPEG compression when JPEG is chosen from the Automatic filter selection (see `AutoFilterGrayImages` on page 38).

`GrayACSImageDict` is based on the `DCTEncode` parameter dictionary described in Section 3.13.3 in the *PostScript Language Reference*.

See "Monochrome (black and white) images" on page 29 for details on how this dictionary is used.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality

Default value

```
<</Qfactor 0.76 /Hsamples [2 1 1 2] /Vsamples [2 1 1 2]>>
```

GrayImageAutoFilterStrategy

Specifies the automatic compression strategy that is used when `AutoFilterGrayImages` is set to `true`. The following table shows the valid values.

Value	UI equivalent	Description
`JPEG`	Automatic (JPEG)	Lossy JPEG compression is used for low-frequency images and lossless Flate compression for high-frequency images.
`JPEG2000`	Automatic (JPEG2000)	Applies only if `CompatibilityLevel` is set to 1.5 or higher. Lossy JPEG2000 compression is used for low-frequency images (images with smooth color changes) and lossless JPEG2000 compression for high-frequency images.

See "Automatic compression" on page 28 for more information.

Supported by

all applications

Type

name

UI name

Compression

Default value

`JPEG`

GrayImageDepth

Specifies the number of bits per sample in the image. The following values are valid:

- The number of bits per sample: `1`, `2`, `4`, or `8`.
- `-1`, which forces the downsampled image to have the same number of bits per sample as the original image

See "Controlling bit depth" on page 31 for more information.

Supported by

Distiller

Type

integer

Default value

`-1`

GrayImageDict

Dictionary of parameters for JPEG compression. `GrayImageDict` is based on the `DCTEncode` parameter dictionary described in Section 3.13.3 in the *PostScript Language Reference*.

See "Monochrome (black and white) images" on page 29 for details on how this dictionary is used.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality

Default value

```
<</Qfactor 0.76 /Hsamples [2 1 1 2] /Vsamples [2 1 1 2]>>
```

GrayImageDownsampleThreshold

Sets the image downsample threshold for grayscale images. This is the ratio of image resolution to output resolution above which downsampling may be performed.

See "Downsampling and subsampling images" on page 29 for details on using this setting.

Supported by

all applications

Type

number

UI name

pixels per inch, for images above: *value* pixels per inch

Default value

```
1.50000
```

GrayImageDownsampleType

Must be one of the following values.

Value	UI equivalent	Description
`Average`	Average Downsampling to	Groups of samples are averaged to get the new downsampled value.
`Bicubic`	Bicubic Downsampling to	Bicubic interpolation is used on a group of samples to get a new downsampled value.
`Subsample`	Subsampling to	The center sample from a group of samples is used as the new downsampled value.
`None`	Distiller: Off CS: Do Not Downsample	No sampling

Supported by

all applications

Type

name

UI name

Sampling

Default value

`Bicubic`

GrayImageFilter

Specifies the compression filter to be used for grayscale images. Ignored if `AutoFilterGrayImages` is `true` or `EncodeGrayImages` is `false`. The following values are valid.

Value	UI equivalent	Description
`DCTEncode`	JPEG	Selects JPEG compression.
`FlateEncode`	ZIP	Selects Flate (ZIP) compression.
`JPXEncode`	JPEG2000	Selects JPEG2000 compression.

Note: JPEG2000 options only appear in UI if `CompatibilityLevel` is set to 1.5 or higher.

`DCTEncode` is used only if the output image has 8 bits per sample; otherwise `FlateEncode` is used. For compatibility with Distiller 3.0 Adobe PDF settings files, Distiller 6.0 and later (as well as Creative Suite applications) revert to Flate compression if this setting is set to `LZWEncode`.

Supported by

all applications

Type

name

UI name

Compression

Default value

`DCTEncode`

GrayImageMinDownsampleDepth

If `DownsampleGrayImages` is `true`, controls the range of bit depths for which gray image downsampling occurs. Valid values are 2, 4 or 8.

For more information, see "Controlling the range of bit depths for which downsampling occurs" on page 30.

Supported by

Distiller

Type

integer

Default value

`2`

GrayImageMinResolution

Imposes a lower limit to the resolution of sampled grayscale images. The legal values are from 9 to 64000, inclusive. How this value is used by Distiller is determined by `GrayImageMinResolutionPolicy`. For more information, see "Specifying a minimum resolution of sampled images" on page 32.

Supported by

Distiller

Type

integer

UI name

Policy->Grayscale Image Policy

Default value

`150`

GrayImageMinResolutionPolicy

Sets the policy for imposition of a lower limit to the resolution of sampled images as specified by the `GrayImageMinResolution` setting. The following values are valid.

Value	UI equivalent	Description
`OK`	Ignore	Distiller's default behavior does not change—i.e., Distiller does not enforce any lower limit on image resolution, ignoring any value specified by `GrayImageMinResolution`.
`Warning`	Warn and continue	A warning is issued every time a sampled grayscale image with resolution smaller than the value specified by `GrayImageMinResolution` is placed in the PDF file. The job continues after issuing the warning.
`Error`	Cancel job	An error occurs when a sampled grayscale image with resolution smaller than the value specified by `GrayImageMinResolution` is placed in the PDF file. The job fails with a limit check error.

For more information, see "Specifying a minimum resolution of sampled images" on page 32.

Supported by

Distiller

Type

name

UI name

Policy->Grayscale Image Policy

Default value

`OK`

GrayImageResolution

Specifies the resolution to which downsampled gray images are reduced. Valid values are 9 to 2400 pixels per inch. A gray image is downsampled if `DownsampleGrayImages` is `true` and the resolution of the input image meets the criteria described in "Downsampling and subsampling images" on page 29.

Supported by

all applications

Type

integer

UI name

pixels per inch

Default value

```
150
```

JPEG2000GrayACSImageDict

Dictionary of parameters for automatic JPEG2000 compression of grayscale. See "JPEG2000" on page 26 for details.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality, Tile Size

Default value

```
<</TileWidth 256 /TileHeight 256 /Quality 15>>
```

JPEG2000GrayImageDict

Dictionary of parameters for JPEG2000 compression of grayscale images. See "JPEG2000" on page 26 for details.

Supported by

all applications

Type

dictionary

UI name

Compression, Image Quality, Tile Size

Default value

```
<</TileWidth 256 /TileHeight 256 /Quality 15 >>
```

Monochrome image settings

This section lists the compression and downsampling settings for monochrome images (images with only one color component and one bit per sample). See "Monochrome (black and white) images" on page 29 for more information.

Note: With the exception of the `AntiAliasMonoImages` and `MonoImageDepth` settings, the compression settings also can be applied to stencil masks created by the `imagemask` operator. Settings behavior is the same in both cases. For details on `imagemask`, see the *PostScript Language Reference*.

AntiAliasMonoImages

If `true`, anti-aliasing is permitted on monochrome images.

Anti-aliasing increases the number of bits per sample in downsampled images to preserve some of the information that is otherwise lost by downsampling. Anti-aliasing is only performed if the image is actually downsampled and `MonoImageDepth` has a value greater than 1. For more information on anti-aliasing see "Controlling bit depth" on page 31.

Note: Distiller does not do anti-aliasing for image masks, regardless of the value of `AntiAliasMonoImages`.

Supported by

Distiller

Type

Boolean

UI name

Anti-Alias to gray

Default value

`false`

CropMonoImages

If `false`, monochrome images are never cropped, whether or not the current clip would remove any image samples.

For more information, see "Disabling of image cropping" on page 33.

Supported by

Distiller

Type

Boolean

Default value

`true`

DownsampleMonoImages

If `true`, monochrome images are downsampled using the resolution specified by `MonoImageResolution`. If `false`, downsampling is not done and the resolution of monochrome images in the PDF file is the same as the source images.

Supported by

all applications

Type

Boolean

UI name

Sampling

Default value

`true`

EncodeMonoImages

If `true`, monochrome images are encoded using the compression filter specified by the value of `MonoImageFilter`. If `false`, no compression filters are applied to monochrome images.

Supported by

all applications

Type

Boolean

UI name

Compression

Default value

`true`

MonoImageDepth

Specifies the number of bits per sample in a downsampled image. Allowed values are:

- The number of bits per sample: `1`, `2`, `4`, or `8`. When the value is greater than `1`, monochrome images are converted to grayscale images.
- `-1`, which forces the downsampled image to have the same number of bits per sample as the original image. (For monochrome images, this is the same as a value of 1.)

`MonoImageDepth` is not used unless `DownsampleMonoImages` and `AntiAliasMonoImages` are `true`. See "Controlling bit depth" on page 31 for more information.

Note: Distiller ignores `MonoImageDepth` for image masks.

Supported by

Distiller

Type

integer

UI name

Anti-Alias to gray

Default value

`-1`

MonoImageDict

Dictionary of parameters for CCITT compression. `MonoImageDict` is based on the `CCITTFaxEncode` parameter dictionary. A value of `-1` for `K` corresponds to CCITT Group 4; 0 corresponds to CCITT group 3. See "Monochrome (black and white) images" on page 29 for more information.

Supported by

all applications

Type

dictionary

UI name

Compression, Quality

Default value

```
<< /K -1 >>
```

MonoImageDownsampleThreshold

Sets the image downsample threshold for monochrome images. This is the ratio of image resolution to output resolution above which downsampling may be performed. See "Downsampling and subsampling images" on page 29 for more information.

Supported by

all applications

Type

number

UI name

pixels-per-inch for images above: *value* pixels-per-inch

Default value

`1.50000` (UI shows 450 pixels-per-inch)

MonoImageDownsampleType

Must be one of the following values.

Value	UI equivalent	Description
`Average`	Average Downsampling to	Groups of samples are averaged to get the new downsampled value.
`Bicubic`	Bicubic Downsampling to	Bicubic interpolation is used on a group of samples to get a new downsampled value.
`Subsample`	Subsampling to	The center sample from a group of samples is used as the new downsampled value.
`None`	Distiller: Off; CS: Do Not Downsample	No sampling

Supported by

all applications

Type

name

UI name

Sampling

Default value

`Bicubic`

MonoImageFilter

Specifies the compression filter to be used for monochrome images. Must be one of the following values:

Value	UI equivalent	Description
`CCITTFaxEncode`	CCITT Group 3/ CCITT Group 4	Selects CCITT Group 3 or 4 facsimile encoding
`FlateEncode`	ZIP	Selects Flate (ZIP) compression
`RunLengthEncode`	Run Length	Selects run length encoding

For compatibility with Distiller 3.0 Adobe PDF settings files, Distiller 6.0 and later (as well as Creative Suite applications) revert to Flate compression if this setting is set to `LZWEncode`. Other values cause Distiller to stop with a range error.

Supported by

all applications

Type

name

UI name

Compression

Default value

`CCITTFaxEncode`

MonoImageMinResolution

Imposes a lower limit to the resolution of sampled monochrome images. The legal values are from 9 to 64000, inclusive. How this value is used by Distiller is determined by `MonoImageMinResolutionPolicy`. For more information, see "Specifying a minimum resolution of sampled images" on page 32.

Supported by

Distiller

Type

integer

UI name

Policy->Monochrome Image Policy

Default value

`300`

MonoImageMinResolutionPolicy

Sets the policy for imposition of a lower limit to the resolution of sampled images as specified by the `MonoImageMinResolution` setting. The following values are valid.

Value	UI equivalent	Description
`OK`	Ignore	Distiller's default behavior does not change—i.e., Distiller does not enforce any lower limit on image resolution, ignoring any value specified by `MonoImageMinResolution`.
`Warning`	Warn and continue	A warning is issued every time a sampled monochrome image with resolution smaller than the value specified by `MonoImageMinResolution` is placed in the PDF file. The job continues after issuing the warning.
`Error`	Cancel job	An error occurs when a sampled monochrome image with resolution smaller than the value specified by `MonoImageMinResolution` is placed in the PDF file. The job fails with a limit check error.

For more information, see "Specifying a minimum resolution of sampled images" on page 32.

Supported by

Distiller

Type

name

UI name

Policy->Monochrome Image Policy

Default value

`OK`

MonoImageResolution

Specifies the resolution to which downsampled monochrome images are reduced; valid values are 9 to 2400 pixels per inch. A monochrome image is downsampled if `DownsampleMonoImages` is `true` and the resolution of the input image meets the criteria described in "Downsampling and subsampling images" on page 29.

Supported by

all applications

Type

integer

UI name

pixels per inch

Default value

`300`

Page Compression Setting

This section describes the page compression setting.

CompressPages

If `true`, Flate compression is used to compress page content streams as well as form, pattern, and Type 3 font content streams.

InDesign also compresses ICC profiles, OutputIntentProfile and shading streams.

Supported by

all applications

Type

Boolean

UI name

Distiller: *none*

InDesign: Compress Text and Line Art

Default value

`true`

Font settings

This section lists the settings available for controlling font embedding and subsetting. For more information on font embedding, see "Using the font settings" on page 33.

Note: Embedding is subject to license; specific fonts can indicate that embedding is not permitted.

AlwaysEmbed

An array consisting either entirely of font names, or of a Boolean value followed by font names. Each name is the PostScript language name of the font (the name given to `definefont`).

- If the array consists entirely of names, Distiller sets its internal list of fonts that must be embedded to be exactly the list of names in the array.
- If the first array value is the Boolean `true`, Distiller adds the font names in the rest of the `AlwaysEmbed` array to its internal list of fonts that must be embedded.
- If the first array value is the Boolean `false`, Distiller removes the font names in the rest of the `AlwaysEmbed` array from its internal list of fonts to be embedded.

If a font name appears in both `AlwaysEmbed` and `NeverEmbed`, `NeverEmbed` takes precedence.

Supported by

Distiller

Type

array

UI name

Always Embed Font

Default value

`[true]`

CannotEmbedFontPolicy

The policy Distiller uses if it cannot find, or cannot embed, the font. The following values are valid.

Value	UI equivalent	Description
`OK`	Ignore	Distiller ignores and continues.
`Warning`	Warn and continue	Distiller displays a warning and continues.
`Error`	Cancel job	Distiller quits distilling the current job.

Supported by

Distiller

Type

name

UI name

When embedding fails

Default value

`Warning`

EmbedAllFonts

If `true`, all fonts that have correct permissions are embedded in the PDF file; if `false`, they are not embedded:

- Creative Suite applications automatically embed fonts.
- Distiller never embeds fonts in its `NeverEmbed` list even if this setting is `true`.

Supported by

Distiller

Name: Boolean

UI name

Embed all fonts

Default value

`true`

EmbedOpenType

If `true`, OpenType fonts are embedded only if all of the following are true:

- The OpenType font is to be embedded within a Type 1 font descriptor
- `EmbedOpenType` is `true`
- `CompatibilityLevel` is 1.6 or higher
- `SubsetFonts` is `false`, or `SubsetFonts` is `true` and the percentage of characters used is greater than `MaxSubsetPct`.

Supported by

Distiller

Type

Boolean

UI name

Embed OpenType fonts

Note: Embed OpenType fonts can only be set from the UI if `CompatibilityLevel` is set to 1.6 or higher.

Default value

`false`

MaxSubsetPct

The maximum percentage of glyphs in a font that can be used before the entire font is embedded instead of a subset. The allowable range is 1 through 100.

Distiller only uses this value if `SubsetFonts` is `true`. For example, a value of 30 means that a font will be embedded in full (not subset) if more than 30% of glyphs are used; a value of 100 means all fonts will be subset no matter how many glyphs are used (because you cannot use more than 100% of glyphs).

Supported by

all applications

Type

integer

UI name

Subset embedded fonts when percent of characters used is less than: *value* %

Default value

```
100
```

NeverEmbed

An array consisting either entirely of font names, or of a Boolean value followed by font names. Each font name must be the PostScript language name of the font (that is, the name given to `definefont`).

- If the array consists entirely of names, Distiller sets its internal list of fonts that must never be embedded to be exactly the list of names in the array.
- If the first array value is the Boolean `true`, Distiller adds the font names in the rest of the `NeverEmbed` array to its internal list of fonts that must never be embedded.
- If the first array value is the Boolean `false`, Distiller removes the font names in the rest of the `NeverEmbed` array from its internal list of fonts to never be embedded.

If a font name appears in both `AlwaysEmbed` and `NeverEmbed`, `NeverEmbed` takes precedence.

Supported by

Distiller

Type

array

UI name

Never Embed Font

Default value

`[true]`

SubsetFonts

If `true`, font subsetting is enabled. If `false`, subsetting is not enabled. Font subsetting embeds only those glyphs that are used in a document, instead of the entire font. This reduces the size of a PDF file that contains embedded fonts. If font subsetting is enabled, the application determines whether to embed the entire font or a subset by the number of glyphs in the font that are used (including component glyphs referenced by 'seac' [Type 1] glyphs), and the value of `MaxSubsetPct`.

Subsetted fonts in the PDF file appear with a 6-letter prefix and a plus (+) sign. For example, Palatino subsetted may appear as:

```
NPBOME+Palatino-Roman
```

Note: Embedded instances of multiple master fonts and of Type 3, TrueType, and CID fonts are always subsetted, regardless of the value of `SubsetFonts`.

Supported by

all applications

Type

Boolean

UI name

Subset embedded fonts when percent of characters used is less than:

Color conversion settings

This section lists the color conversion settings supported by Distiller. Some are indicated as supported by Creative Suite applications and can approximate the settings defined in the `CreativeSuite` namespace, which provide greater control over color conversions. Creative Suite applications can use these settings if present and write them to settings files for compatibility. See "Using the color conversion settings" on page 34 for details.

CalCMYKProfile

The name of the ICC profile that is used for tagging or converting CMYK images, text, and/or graphics.

Note: Creative Suite applications give precedence to their own color management settings and may use this setting for compatibility with Distiller; see "Using the color conversion settings" on page 34 for details.

Supported by

all applications

Type

string

UI name

Working Spaces: CMYK

Default value

```
(U.S. Web Coated SWOP v2)
```

CalGrayProfile

The name of the ICC profile that is used for tagging or converting Gray images, text, and/or graphics.

Note: Creative Suite applications give precedence to their own color management settings and may use this setting for compatibility with Distiller; see "Using the color conversion settings" on page 34 for details.

Supported by

all applications

Type

string

UI name

Working Spaces: Gray

CalRGBProfile

The name of the ICC profile that is used for tagging or converting RGB images, text, and/or graphics.

Note: Creative Suite applications give precedence to their own color management settings and may use this setting for compatibility with Distiller; see "Using the color conversion settings" on page 34 for details.

Supported by

all applications

Type

string

UI name

Working Spaces: RGB

Default value

```
(sRGB IEC61966-2.1)
```

ColorConversionStrategy

Sets the color conversion strategy, which includes the output color family and color space and the inclusion of ICC profiles.

See "Using the color conversion settings" on page 34 for details on how to use this setting.

The section "Color settings interchange" on page 37 describes how the values of this setting map to those in the `CreativeSuite namespace`.

The following values are valid.

Value	UI equivalent
`LeaveColorUnchanged`	Leave Color Unchanged
`UseDeviceIndependentColor`	Tag Everything for Color Management (no conversion)
`UseDeviceIndependendColor-ForImages`	Tag Only Images for Color Management (no conversion)
`sRGB`	Convert All Colors to sRGB
`CMYK`	Convert All Colors to CMYK

Supported by

all applications

Type

name

UI name

Distiller: Color Management Policies

Creative Suite: no single setting (see below)

Default value

`sRGB`

ColorSettingsFile

Specifies the name of a color settings file to be used for color conversions. When this file is specified as a non-empty value, all other color conversion settings are ignored and not selectable in the UI.

If the name is `(None)` or `()`, the other settings are used.

The Creative Suite applications use color settings files but do not use this setting. See "Using the color conversion settings" on page 34 for more information.

Supported by

Distiller

Type

string

UI name

Settings File

Default value

`()`

DefaultRenderingIntent

Specifies the rendering intent for objects to be written to the PDF document, when not otherwise specified in the PostScript job by means of the `findcolorrendering` and `setcolorrendering` operators (see Section 7.1.3 in the *PostScript Language Reference* for details).

If the value of this setting is `Default`, no rendering intent is written to the PDF document. The following values are valid.

Value	UI equivalent
`Default`	Preserve
`Perceptual`	Perceptual
`Saturation`	Saturation
`AbsoluteColorimetric`	Absolute Colorimetric
`RelativeColorimetric`	Relative Colorimetric

Supported by

Distiller

Type

name

UI name

Document Rendering Intent

Default value

`Default`

ParseICCProfilesInComments

If `true`, Distiller honors EPS embedded ICC profiles when distilling. ICC profiles are honored only if they are enclosed in two DSC pairs: `ICCProfile` and `SetColorSpace`. See the ICC specification (available at http://www.color.org), section B.2, for details on the syntax of these comment pairs.

This setting is ignored if `CompatibilityLevel` is set to 1.2.

Supported by

Distiller

Type

Boolean

UI name

none

Default value

`true`

PreserveDICMYKValues

Describes what to do with color values for device-independent CMYK color spaces. This setting is used only if `ColorConversionStrategy` is `CMYK`.

If `true`, CIEBasedDEFG CMYK color values are treated as DeviceCMYK values; CIEBasedDEFG color spaces will be ignored and discarded. If `false`, a conversion from CIEBasedDEFG color space to CMYK working space is performed.

Supported by

Distiller

Type

Boolean

UI name

Preserve CMYK values for calibrated CMYK color spaces

Default value

`true`

PreserveHalftoneInfo

If `true`, Distiller passes halftone screen information (frequency, angle, and spot function) into the PDF file. If `false`, halftone information is not passed in.

Supported by

Distiller

Type

Boolean

UI name

Preserve Halftone Information

Default value

`false`

sRGBProfile

(Read Only) The name of the ICC profile that is used for converting device-dependent or device-independent color spaces to CalRGB (PDF 1.2) or ICCBased (PDF 1.3 and above).

Supported by

all applications

Type

string

Default value

`(sRGB IEC61966-2.1)`

TransferFunctionInfo

Determines how Distiller handles transfer functions, which are traditionally used to compensate for dot gain or dot loss that may occur when an image is transferred to film. For example, a file that is intended for output on a particular imagesetter may contain transfer functions that compensate for the dot gain inherent with that printer. The following values are valid.

Value	UI equivalent	Description
`Preserve`	Preserve	Distiller preserves (passes into the PDF file) transfer functions.
`Remove`	Remove	Distiller ignores transfer functions. They are neither applied to the color values by Distiller nor passed into the PDF file.
`Apply`	Apply	Distiller uses the transfer function to modify the data it writes to the PDF file, instead of writing the transfer function itself to the file. This value is ignored by Distiller 4.0 but supported by Distiller 5.0 and later. It is sometimes used to achieve artistic effects (although the *PostScript Language Reference* discourages such usage).

If you are generating PDF/X-compliant files, do not set this to `Preserve`.

Supported by

Distiller

Type

name

UI name

When transfer functions are found:

Default value

`Preserve`

UCRandBGInfo

Tells Distiller whether to pass the arguments to `setundercolorremoval` and `setblackgeneration` into the PDF file.

Must be one of the following values.

Value	UI equivalent	Description
`Preserve`	checked	Distiller preserves (passes into the PDF file) the arguments.
`Remove`	unchecked	Distiller ignores the arguments.

See Section 7.2.3 in the *PostScript Language Reference* for details on the `setundercolorremoval` and `setblackgeneration` operators and descriptions of undercolor removal (UCR) and black generation (BG)

Supported by

Distiller

Type

name

UI name

Preserve Under Color Removal and Black Generation

Default value

`Remove`

Advanced Adobe PDF settings

This section lists the settings in the Advanced panel of the Distiller settings dialog box. Some of these settings are also supported by Creative Suite applications, as indicated.

AllowPSXObjects

If `true`, allow PostScript XObjects. For a description of PostScript XObjects, see the *PDF Reference*.

Supported by

Distiller

Type

Boolean

UI name

Allow PostScript XObjects

Default value

`true`

AllowTransparency

The `SetTransparency pdfmark` is a `pdfmark` extension used to produce transparency in PDF 1.4 and later. For more details, see the pdfmark Reference. If this setting is `true`, `[... /SetTransparency pdfmark` is allowed in PS jobs if `CompatibilityLevel` is 1.4 or higher. If `false`, this `pdfmark` is treated as error.

Note: Creative Suite applications always include transparency in the exported PDF if `CompatibilityLevel` is 1.4 or higher.

Supported by

Distiller

Type

Boolean

Default value

`false`

ASCII85EncodePages

If `true`, Distiller ASCII85-encodes binary streams such as page content streams, sampled images, and embedded fonts, resulting in a PDF file that is pure ASCII. If `false`, Distiller does not encode the binary streams, resulting in a PDF file that may contain substantial amounts of binary data. Distiller checks the value of this setting only once per document. Any change to it must be made before any marks are placed on the first page of the document.

Supported by

Distiller

Type

Boolean

Default value

`false`

AutoPositionEPSFiles

If `true`, Distiller resizes the created page to the size of the EPS file using the `%%BoundingBox` comment in the header of the file, and centers the EPS file on the page when the EPS file is distilled. Distiller ignores this setting if `ParseDSCComments` is `false`.

Supported by

Distiller

Type

Boolean

UI name

Resize page and center artwork for EPS files

Default value

`true`

CreateJDFFile

If `true`, Distiller produces a Job Definition Format (JDF) file that reflects the parameters used for distillation. If `false`, a JDF file is not produced.

The Job Definition Format (JDF) Specification is owned and maintained by the International Cooperation for the Integration of Processes in Prepress, Press and PostPress (CIP4) (www.cip4.org). Distiller 7.0 complies with JDF Specification Version 1.1 Revision A, published on September 5, 2002. It is available on the web at:

http://www.cip4.org/documents/jdf_specifications/index.html

Note: The Acrobat 7 Professional product now supports creation of both JDF 1.1- and JDF 1.2-compliant JDF files. For more information, see the *Acrobat Guide* in Distiller online Help.

The Adobe Normalizer product (see *Using Adobe Normalizer Server, Version 6.0.4*) is also capable of producing JDF files, but it can consume them as well. "Conversions Related to JDF" on page 159," describes how Normalizer interprets and converts Distiller parameters; use this information to understand the JDF file created by Distiller.

The JDF file is output to the current directory with the `.jdf` extension. The filename is the same as the `.log` file and the file that is being distilled. (The "current directory" is the directory where the new PDF file is output.)

Note: The `JDF` pdfmark allows the PostScript file/stream being distilled to specify certain elements and attributes to be added to a JDF file. For details, see *Using Adobe Normalizer Server, Version 6.0.4* and pdfmark Reference.

Supported by

Distiller

Type

Boolean

UI name

Create Job Definition Format (JDF) file

Default value

`false`

CreateJobTicket

If `true`, Distiller creates a Job Ticket object in the PDF file that contains specific information about this file—such as trapping information—that can be passed along to another application or print device.

This setting pertains to Portable Job Ticket Format 1.1, as described in *Portable Job Ticket Format, version 1.1* (Technical Note #5620).

Supported by

Distiller

Type

Boolean

UI name

Save Portable Job Ticket inside PDF file

Default value

`false`

DetectBlends

If `true`, Distiller enables the conversion of PostScript gradients to smooth shades. If `false`, Distiller disables conversion.

Note: If `CompatibilityLevel` is less than 1.3, Distiller disables conversion regardless of the value of `DetectBlends`. In addition, Distiller always disables conversion if idiom recognition is turned off in the prologue file or in the PostScript file itself.

Distiller uses two methods to perform the conversion of gradients to smooth shades:

- The PostScript Language Level 3 feature called *idiom recognition* replaces certain procedures (idioms) with others having equivalent behavior but producing better quality results. (See "Idiom Recognition" on page 119 of the *PostScript Language Reference* for details.) `DetectBlends` enables the subset of idioms that detect gradients (or blends) for the following applications: Adobe Illustrator, Adobe Freehand®, CorelDraw, and QuarkXPress.
- Distiller also converts gradients to smooth shades independently of idiom recognition. This method is application-independent, but it is less reliable than the first.

In Distiller 4.0, the blend detecting idioms (first method) was controlled by the `IdiomRecognition` PostScript feature, while the second method was controlled by `DetectBlends`. You had to turn off `IdiomRecognition` to use `DetectBlends`.

In Distiller 5.0 and above, `DetectBlends` controls the blend detecting idioms. By default `IdiomRecognition` is turned on in Distiller 5.0 and above, and the blend detecting idioms are controlled using `DetectBlends`. You can still use the PostScript feature `IdiomRecognition` with the `setuserparams` operator, if needed.

Supported by

Distiller

Type

Boolean

UI name

Convert gradients to smooth shades

Default value

`true`

DetectCurves

The value of this setting must be in the range from 0.0000 to 10.0000. If the value is 0.0000, the feature is disabled. If positive, Distiller investigates graphics for curves that are not described efficiently and thus result in unacceptably large file sizes. Distiller converts these curves into bezier curves that take up much less file space.

This setting represents a value in user space (72 dpi) that controls Distiller's curve-fitting algorithm: the curve-fitting results should not part from the original line segments by more than this number. Visual inspection of the results suggests that the 0.1000 value yields the closest approximation to the original curve.

Supported by

Distiller

Type

number

UI name

Convert smooth lines to curves

Default value

`0.1`

DSCReportingLevel

Level can be either `0`, `1`, or `2`. `0` means no additional reporting. Level 1 shows all input as it is parsed and shows a tree crawl when getting into bad states. Level 2 shows transitions in addition to the information in Level 1.

Supported by

Distiller

Type

integer

Default value

`0`

EmbedJobOptions

If `true`, the settings file used to create the PDF is embedded in the PDF file and is accessible through the Acrobat UI (the Attachments tab in Acrobat 7). In the PDF file, the Adobe PDF creation settings file becomes an item in the `Names->EmbeddedFiles` tree (see the *PDF Reference*).

Note: Applications other than Distiller do not embed the settings file, regardless of the value of this setting.

Supported by

Distiller

Type

Boolean

UI name

Distiller: Save Adobe PDF settings inside PDF file

Default value

`false`

EmitDSCWarnings

If `true`, Distiller displays warning messages about questionable or incorrect DSC comments during the distillation of the PostScript file. Distiller ignores this setting if `ParseDSCComments` is `false`.

Supported by

Distiller

Type

Boolean

UI name

Log DSC warnings

Default value

`false`

LockDistillerParams

If `true`, Distiller ignores any settings specified by `setdistillerparams` operators in the incoming PostScript file and uses only those settings present in the Adobe PDF settings file (or their default values if not present).

If `false`, any settings specified in the PostScript file override the initial settings. These settings are in effect for the duration of the current `save` level.

Note: There are a number of settings whose values cannot be changed by `setdistillerparams` in a PostScript file. See "Modifying settings during the job" on page 21 for a complete list.

Supported by

Distiller

Type

Boolean

UI name

Allow PostScript file to override Adobe PDF settings

Default value

`false`

OPM

Controls the overprint mode strategy in the job. Set to 0 for full overprint or 1 for non-zero overprint. For more information, refer to Technical Note #5044, *Color Separation Conventions for PostScript Language Programs*, and the *PDF Reference*.

Note: Distiller ignores this setting if `PreserveOverprintSettings` is `false`.

Supported by

Distiller

Type

integer

UI name

Overprinting default is nonzero overprinting

Default value

`1`

ParseDSCComments

If `true`, Distiller parses the DSC comments for any information that might be helpful for distilling the file or for information that is passed into the PDF file. If `false`, Distiller treats the DSC comments as pure PostScript comments and ignores them.

Supported by

Distiller

Type

Boolean

UI name

Process DSC Comments

Default value

`true`

ParseDSCCommentsForDocInfo

If `true`, Distiller attempts to preserve the Document Information from the PostScript DSC comments as properties of the PDF document. The following table lists this information.

Document Information	Source
Author	from DSC keyword: `%%For:`
Creator	from DSC keyword: `%%Creator:`
Title	from DSC keyword: `%%Title:`
Producer	from Distiller product name ("Acrobat Distiller 7.0")
CreationDate	from Distiller time stamp (creation time of PDF file)
ModDate	from Distiller time stamp (creation time of PDF file)

Note: Distiller ignores this setting if `ParseDSCComments` is `false`.

Distiller 4.0 and higher places the Document Information in the `Info` dictionary of the PDF file; you can view the information in Acrobat by clicking File > Document Properties. Starting with version 5, Distiller also embeds the Document Information as XML in the PDF file. To embed the information, Distiller adds a `Metadata` key in the Catalog dictionary whose value is an indirect reference to a metadata stream object. The metadata object contains the metadata (the Document Information) for the PDF document. The metadata is represented as RDF, in conformance with Adobe's Extensible Metadata Platform (XMP).

Note: If `true`, document properties of Microsoft Office files are carried into the PDF. A setting of `false` prevents this transfer of information.

Supported by

Distiller

Type

Boolean

UI name

Preserve document information from DSC

Default value

`true`

PassThroughJPEGImages

If `true`, JPEG images (images that are already compressed with the `DCTEncode` filter) are "passed through" Distiller without re-compressing them. (Distiller does perform a decompression to ensure that images are not corrupt, but then passes the original compressed image to the PDF file.) Images that are not compressed will still be compressed according to the image settings in effect for the type of image (for example, `ColorImageFilter`, etc.).

If `false`, all JPEG encoded images are decompressed and recompressed according the compression settings in effect.

Note, however, that JPEG images that meet the following criteria are *not* passed through even if the value of `PassThroughJPEGImages` is `true`:

- The image will be downsampled.
- `ColorConversionStrategy` is `sRGB` and the current PostScript color space (for the image) is not `DeviceRGB` or `DeviceGray`.
- The image will be cropped—i.e., the clip path is such that more than 10% of the image pixels will be removed.

Creative Suite applications do not use this setting. However, Illustrator and InDesign normally behave as if it were `true` with regard to placed PDF files containing compressed images. That is, they do not normally uncompress and recompress them, unless color conversion or downsampling takes place.

Passing through JPEG images has these advantages:

Performance: Only decompression and not recompression occurs.

No loss of image data: DCT encoding inherently causes some loss of data; thus, with this option, since no recompression occurs, no data is lost.

No loss of metadata: When Distiller decompresses an image, all metadata is discarded; thus, with this option, no metadata is lost since no recompression on the decompressed image occurs.

Note: Distiller's hard-coded default value of `PassThroughJPEGImages` is `false` to conform to older settings files where this setting did not exist. Most predefined Adobe PDF settings files set it to `true`. The `Smallest File Size` settings file sets it to `false` since that generally results in smaller file sizes. However, in some cases this setting could actually increase file size, for example, if the original JPEG in the PostScript file was compressed with a `Quality` setting lower than the `Quality` setting in the settings file.

Supported by

Distiller

Type

Boolean

UI name

Save original JPEG images in PDF if possible

Default value

`false`

PreserveCopyPage

Default value

`true`

If `true`, Distiller maintains PostScript Language Level 2 compatibility for the `copypage` operator. If `false`, Distiller uses the PostScript Language Level 3 definition of the `copypage` operator. See the *PostScript Language Reference* for more information.

Supported by

Distiller

Type

Boolean

UI name

Preserve Level 2 copypage semantics

PreserveEPSInfo

If `true`, and `ParseDSCComments` is `true`, Distiller attempts to preserve the encapsulated PostScript (EPS) information in the PostScript file as properties of the resulting PDF file. The distilled EPS content is identified as marked content using the `EmbeddedDocument` key.

The following table lists this information.

Document Information	Source
Author	from DSC keyword: `%%For:`
Creator	from DSC keyword: `%%Creator:`
Title	from DSC keyword: `%%Title:`

Starting with version 5, Distiller also embeds the information for embedded EPS files as XML in the PDF file. To do this, Distiller performs the following tasks:

- Adds a `Metadata` key in the property list of the marked content container for the EPS.
- Stores the property list as an indirect reference in the page resources object.

The value of the `Metadata` key is an indirect reference to the metadata stream object, which contains the metadata (the EPS information). The metadata is represented as RDF, in conformance with Adobe's XMP.

Supported by

Distiller

Type

Boolean

UI name

Preserve EPS information from DSC

Default value

`false`

PreserveFlatness

If `true`, the PostScript flatness set by the `setflat` operator is preserved. If `false`, flatness is discarded. Preserving flatness can increase rendering and printing speeds, since less time is spent determining how to precisely render curves and circles.

Supported by

Distiller

Type

Boolean

Default value

`true`

PreserveOPIComments

If `true`, Distiller places the page contents within a set of Open Prepress Interface (OPI) comments in a Form XObject dictionary and preserves the OPI comment information in an OPI dictionary attached to the Form. Page contents data within a set of OPI comments may include proxy images, high-resolution images, or nothing.

If `PreserveOPIComments` is `false`, Distiller ignores OPI comments and their page contents. Setting `PreserveOPIComments` to `false` results in slightly simpler and smaller PDF files. Doing so is acceptable when use of an OPI server is not anticipated.

Distiller ignores `PreserveOPIComments` if `ParseDSCComments` is `false`.

Distiller recognizes both OPI 1.3 and OPI 2.0. See the specifications for OPI 1.3 and 2.0 (TN #5660) at the Acrobat Developer Center.

Note: Creative Suite applications (Illustrator and InDesign) always behave as though this setting were `true`.

Supported by

Distiller, InDesign

Type

Boolean

UI name

Preserve OPI comments

Default value

`false`

PreserveOverprintSettings

If `true`, Distiller passes the value of the `setoverprint` operator through to the PDF file. If `false`, overprint is ignored (the information is not passed.

For Illustrator, this setting is relevant only when saving to PDF 1.3 (no transparency support) and the document contains overprint. If this setting is `true`, any overprint in the artwork is passed through to the PDF file unchanged. If `false`, any overprint in the artwork is stripped out and the resulting PDF file will not contain any.

Note: InDesign uses the `SimulateOverprint` setting that has additional options.

Supported by

Distiller, Illustrator

Type

Boolean

UI name

Distiller: Preserve overprint settings

Illustrator: Overprints: Preserve (Advanced panel)

Default value

`true`

UsePrologue

If `true`, Distiller uses the `prologue.ps` file in the `Data` subdirectory and distills it prior to any PostScript job that is sent through. Distiller also distills the `epilogue.ps` file in the same directory after the same PostScript job is run. You can add any legal PostScript code and comments to these two files.

Supported by

Distiller

Type

Boolean

UI name

Use Prologue.ps and Epilogue.ps

Default value

`false`

Standards settings

This section lists the settings related to PDF/X and PDF/A compliance. See "Using the standards settings" on page 42 for more information.

CheckCompliance

Specifies the name of a standard to which the PDF file should comply. The value is an array of names; however, at the present time, only one name may be specified. The following table shows the valid values.

Value	UI equivalent (Distiller)
`None`	None
`PDF/A-1b:2005 (CMYK)`	PDF/A (Acrobat 5.0 Compatible)
`PDFX1a:2001`	PDF/X-1a (Acrobat 4.0 Compatible)
`PDFX1a:2003`	PDF/X-1a (Acrobat 5.0 Compatible)
`PDFX3:2002`	PDF/X-3 (Acrobat 4.0 Compatible)
`PDFX3:2003`	PDF/X-3 (Acrobat 5.0 Compatible)

Note: The Creative Suite UI uses the standards names in the first column. Creative Suite applications do not support PDF/A.

In Distiller 7 and the Creative Suite, this setting takes precedence over `PDFX1aCheck` and `PDFX3Check`. For more information, see "Using the compliance checking settings" on page 43.

For Creative Suite applications, `CheckCompliance` also takes precedence over values of other settings (such as `ColorConversionStrategy` or `CompatibilityLevel`) that could potentially result in a file that is not compliant, and the values of the other settings are adjusted accordingly.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

array

UI name

Distiller: Compliance Standard

Creative Suite: Standard

Default value

`[/None]`

PDFX1aCheck

If `true`, checks compliance with the PDF/X-1a standard (ISO 15930-1:2001) and a PDF/X compliance report is written to the message log.

A value of `/PDFX1aCheck true` is equivalent in Distiller 7 to a value of `/CheckCompliance [ /PDFX1a:2001 ]`.

Note: This setting is retained for compatibility with Distiller 6. For more information on how this setting works, see "Using the standards settings" on page 42.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

Boolean

UI name

none

Default value

`false`

PDFX3Check

If `true`, checks compliance with the PDF/X-3 standard (ISO 15930-3:2002) and a PDF/X compliance report is written to the message log. A value of `/PDFX3Check true` is equivalent in to a value of `/CheckCompliance [ /PDFX3:2002 ]`.

Note: This setting is retained for compatibility with Distiller 6. For more information on how this setting works, see "Using the standards settings" on page 42.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

Boolean

UI name

none

Default value

`false`

PDFXBleedBoxToTrimBoxOffset

If the `BleedBox` entry is not specified in the page object, `BleedBox` is set to `TrimBox` with offsets. Offsets are specified as `[ left right top bottom ]`. All numbers must be greater than or equal to 0.0. `BleedBox` offsets place the `BleedBox` entirely outside the `TrimBox`.

Note: This setting is ignored if `PDFXSetBleedBoxToMediaBox` is `true`.

Supported by

Distiller

Type

array

UI name

Set BleedBox to TrimBox with offsets (Points)

Default value

```
[0.00000 0.00000 0.00000 0.00000]
```

PDFXCompliantPDFOnly

Determines what to do when PDF/X compliance tests are not passed. The following table shows the valid values.

Value	UI equivalent	Description
`true`	Cancel job	A PDF document is produced only if PDF/X compliance tests are passed.
`false`	Continue	A PDF document is produced regardless of whether PDF/X compliance tests are passed. Distiller does not insert PDF/X additional key/value pairs into the created PDF file.

Note: InDesign uses its `ErrorControl` setting to determine what to do in this situation.

Supported by

Distiller

Type

Boolean

UI name

When not compliant

Default value

`false`

PDFXNoTrimBoxError

If `true` and both the `TrimBox` and `ArtBox` entries are not specified in the page object, the condition is reported as an error.

Supported by

Distiller

Type

Boolean

UI name

Report as error

Default value

`true`

PDFXOutputCondition

This setting provides an optional comment which, if present, is added to the PDF file and describes the intended printing condition in a form that should be meaningful to a human operator at the site receiving the PDF document.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

string

UI name

Distiller: Output Condition

Creative Suite: Output Condition Name

Default value

`()`

PDFXOutputConditionIdentifier

This setting is a reference name that is specified by the output intent profile name registry. The entry is automatically entered for known output intent profile names.

For Distiller only, if the value of `PDFXOutputIntentProfile` is `(Use Output Condition Identifier)`, this setting must be provided for PDF/X validation to succeed.

For a description of how this setting is used to fill out entries in the PDF/X output intent dictionary, see "Using the PDF/X output intent settings" on page 44.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

string

UI name

Output Condition Identifier

Default value

`()`

PDFXOutputIntentProfile

This setting indicates the characterized printing condition for which the document has been prepared and is required for PDF/X compliance. The value may be one of the following:

- The name of a specific profile. The UI lists a number of available profiles. This is the only value supported by Creative Suite applications.
- `(Use Output Condition Identifier)` The profile is specified by `PDFXOutputConditionIdentifier`. This value is used by Distiller only.
- `(None)` This value should be used for workflows that require the output intent dictionary to be specified in the PostScript document and that require compliance checking to fail if it is not specified. In the UI, it corresponds to "No Default Profile". This value is used by Distiller only.

For a description of how these values are used to fill out entries in the PDF/X output intent dictionary, see "Using the PDF/X output intent settings" on page 44.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

string

UI name

Output Intent Profile Name

Default value

`()`

PDFXRegistryName

This setting specifies a URL where more information regarding the output intent profile can be obtained. This entry is automatically populated for recognized ICC profile names. If the value of `PDFXOutputIntentProfile` is `(Use Output Condition Identifier)`, this setting must be provided for PDF/X validation to succeed.

Supported by

Distiller, Illustrator, InDesign, Photoshop

Type

string

UI name

Registry Name (URL)

Default value

`()`

PDFXSetBleedBoxToMediaBox

If `true` and the `BleedBox` entry is not specified in the page object, `BleedBox` is set to `MediaBox`.

Supported by

Distiller

Type

Boolean

UI name

Set BleedBox to MediaBox

Default value

`true`

PDFXTrapped

Indicates the state of trapping within the file. Can be one of the following values:

Value	UI equivalent	Description
`Unknown`	Leave Undefined	The trapped state indicated in the PostScript file is used if present; otherwise the state is undefined.
`False`	Insert False	The document is not trapped.
`True`	Insert True	The document is trapped.

Note: `True` and `False` are name objects, not the Boolean values `true` and `false`.

A value of `True` or `False` is required for PDF/X compliance. If the PostScript file does not specify that the document is trapped, then the value provided here is used. `Unknown` should be used for workflows that require that the document specify a trapped state and for which compliance checking should fail if it is not present in the document.

Note: Illustrator can set this value to `True` or `False` via "Mark as Trapped." InDesign always specifies `False` for PDF/X compliant files.

Supported by

Distiller, Illustrator

Type

name

UI name

Distiller: Trapped

Illustrator: Marked as Trapped

Default value

`False`

PDFXTrimBoxtoMediaBoxOffset

If both the `TrimBox` and `ArtBox` entries are not specified in the page object, `TrimBox` is set to `MediaBox` with offsets. Offsets are specified as [*`left right top bottom`*]. All numbers must be greater than or equal to 0.0. `TrimBox` offsets place `TrimBox` entirely inside `MediaBox`.

Note: This setting is ignored if `PDFXNoTrimBoxError` is `true`.

Supported by

Distiller

Type

array

UI name

Set TrimBox to MediaBox with offsets *(units)*

Default value

```
[0.00000 0.00000 0.00000 0.00000]
```

4 Other Namespaces

This chapter describes the settings that belong to namespaces other than the `Common` namespace. See "Namespaces" on page 11 for information on how namespaces are organized.

CreativeSuite namespace settings

The following settings belong to the `CreativeSuite` namespace. They are supported by one or more of the applications in Creative Suite version 2.0 and later (Illustrator, InDesign, and Photoshop). The entries for each setting indicate which applications support it.

AddBleedMarks

If `true`, bleed marks are included in the PDF file.

Supported by

InDesign

Type

Boolean

UI name

Add Bleed Marks

Default value

`false`

AddColorBars

If `true`, color bars are included in the PDF file.

Supported by

Illustrator, InDesign

Type

Boolean

UI name

Color Bars

Default value

`false`

AddCropMarks

If `true`, crop marks are included in the PDF file.

Supported by

Illustrator, InDesign

Type

Boolean

UI name

InDesign: Crop Marks

Illustrator: Trim Marks

Default value

`false`

AddPageInfo

If `true`, page information (document name, page number, and time stamp) are included in the PDF file.

Supported by

Illustrator, InDesign

Type

Boolean

UI name

Page Information

Default value

`false`

AddRegMarks

If `true`, registration marks are included in the PDF file.

Supported by

Illustrator, InDesign

Type

Boolean

UI name

Registration Marks

Default value

`false`

BleedOffset

An array of four floating point numbers with values between 0.0 to 432.0, representing the offsets of the bleed in points from the top, left, bottom, and right edges of the document.

In InDesign, if `UseDocumentBleed` is `true`, the bleed values are taken from the document bleed rather than the value of this setting.

Supported by

Illustrator, InDesign

Type

array

UI name

Bleeds

Default value

```
[0.0, 0.0, 0.0, 0.0]
```

ConvertColors

Specifies whether colors should be converted to CMYK or RGB. The following table shows the valid values.

Value	UI equivalent	Description
`NoConversion`	No Color Conversion	Colors are not converted.
`ConvertToCMYK`	*See below*	Colors are converted to CMYK
`ConvertToRGB`	*See below*	Colors are converted to RGB.

`ConvertToCMYK` or `ConvertToRGB` do not map precisely to the UI options, which are "Convert to Destination" or "Convert to Destination (preserve numbers)". The "destination" referred to is a specified by the appropriate profile (CMYK or RGB) and additional settings. See "Using the color conversion settings" on page 34 for more details.

Note: When converting to CMYK, the resulting Common settings are supported by Distiller 7.0, but previous versions of Distiller did not support conversion to CMYK.

Supported by

Illustrator, InDesign, Photoshop

Type

name

UI name

Color Conversion

Default value

`NoConversion`

DestinationProfileName

When the value of `DestinationProfileSelector` is `UseName`, this setting specifies the name of the destination profile to be used.

When the value of `DestinationProfileSelector` is one of the document or working space profiles, Creative Suite applications store the profile name in this setting when saving the settings. This enables determining whether a setting has been changed outside the application. For example, when converting to CMYK:

- If the value of this setting is a profile name, it should be the same as that of `CalCMYKProfile`. Different profile names imply that `CalCMYKProfile` was changed in Distiller; therefore, Creative Suite applications will use `CalCMYKProfile` instead of `DestinationProfileName`.
- If the value of this setting is the empty string, Creative Suite applications will use the document CMYK profile regardless of the value of `CalCMYKProfile`.

Supported by

Illustrator, InDesign, Photoshop

Type

string

UI name

Destination

DestinationProfileSelector

Specifies which destination color profile should be used. The following table shows the valid values.

Value	UI equivalent	Description
`NA`	N/A	No profile
`UseName`	*Any named profile*	The profile specified by `DestinationProfileName` is used.
`WorkingCMYK`	Working CMYK	The effective working CMYK profile is used. The profile name is also written to `DestinationProfileName`.
`WorkingRGB`	Working RGB	The effective working RGB profile is used.
`DocumentCMYK`	Document CMYK	The effective document CMYK profile is used. The profile name is also written to `DestinationProfileName`.
`DocumentRGB`	Document RGB	The effective document RGB profile is used.

Supported by

all applications

Type

name

UI name

Destination

Default value

`NA`

Downsample16BitImages

If `true`, 16 bits-per-channel images are converted to 8 bits-per-channel images. If `false`, 16-bit images are not converted, and Flate (ZIP) is the only compression method available.

This setting is available only if `CompatibilityLevel` is 1.5 or greater. For 1.4 or earlier, 16-bit images are always converted to 8 bits per channel.

Supported by

Photoshop

Type

Boolean

UI name

Convert 16 Bit/Channel Image to 8 Bit/Channel

Default value

`true`

FlattenerPreset

A dictionary that specifies a preset or set of options for flattening transparency. One of the entries is `PresetSelector`, which is a string corresponding to a preset file. It can correspond to one of the preset files that specify transparency flattening: `LowResolution`, `MediumResolution`, or `HighResolution`. It can also take the value `UseName`, which means that the `PresetName` entry specifies the name of the preset file.

The other entries are: `ClipComplexRegions`, `ConvertStrokesToOutlines`, `ConvertTextToOutlines`, `GradientResolution`, `LineArtTextResolution`, and `RasterVectorBalance`. See Creative Suite application Help for more information on these transparency flattening options.

Note: This setting applies only when `CompatibilityLevel` is 1.3; that is, for PDF versions where transparency is not supported.

Supported by

Illustrator, InDesign

Type

dictionary

UI name

Transparency Flattener: Preset

Default value

```
<</PresetSelector /MediumResolution>>
```

GenerateStructure

If `true`, a Tagged PDF document is generated. Tagged PDF contains information about the logical structure of the document that can be used to provide document reflow, improved accessibility, and interchange with other file formats. See the *PDF Reference* for more information on Tagged PDF.

If `CompatibilityLevel` is 1.5 or later (that is, Acrobat 6 or later compatibility is chosen), tags are compressed for smaller file size.

Supported by

InDesign

Type

Boolean

UI name

Create Tagged PDF

Default value

`false`

IncludeBookmarks

A Boolean value to include bookmarks, defined in an InDesign document, in generated PDF.

Supported by

InDesign

Type

Boolean

UI name

Bookmarks

Default value

`false`

If `true`, bookmarks are included in generated PDF.

IncludeHyperlinks

A Boolean value to include hyperlinks in generated PDF.

Supported by

InDesign

Type

Boolean

UI name

Hyperlinks

Default value

`false`

If `true`, hyperlinks are included in generated PDF.

IncludeInteractive

If `true`, interactive elements should be included in the PDF document. These can be:

Multimedia elements: If `true`, the `MultimediaHandling` setting controls how the elements should be included.

Supported by

InDesign

Type

Boolean

UI name

Interactive Elements

Default value

`false`

IncludeLayers

If `true`, layers (optional content) are included in generated PDF; available in PDF 1.5 or later.

Supported by

Illustrator, InDesign

Type

Boolean

UI name

Create Acrobat Layers

Default value

`false`

IncludeProfiles

If `false`, color profiles are not included in generated PDF. This corresponds to Don't Include Profiles in the UI. If `true`, which profiles are included is dependent on the value of other settings. See "Creative Suite color conversion settings" on page 36 for details.

Supported by

Illustrator, InDesign, Photoshop

Type

Boolean

UI name

Profile Inclusion Policy

Default value

`false`

MarksOffset

A floating-point number between `0.0` and `6.0` indicating the printer offset in points.

Supported by

Illustrator, InDesign

Type

number

UI name

Offset

Default value

`0`

MarksWeight

A floating-point number between `0.125` and `0.5` indicating the line weight in points to be used for printer's marks.

Supported by

Illustrator, InDesign

Type

number

UI name

InDesign: Weight

Illustrator: Trim Mark Weight

Default value

`0.25`

MultimediaHandling

Specifies how multimedia objects (movies and sounds) are referenced in the PDF file. This setting is relevant only if `IncludeInteractive` is `true`. Valid values are:

Value	UI equivalent	Description
`UseObjectSettings`	Automatic	If `CompatibilityLevel` is PDF 1.5 or later, all objects are embedded. If it is PDF 1.4 or earlier, all objects are linked. **Note:** Used by InDesign only.
`LinkAll`	Link All	All objects are linked when `CompatibilityLevel` is PDF 1.5 or later; otherwise, sound clips are embedded and movies are linked.
`EmbedAll`	Embed All	All objects are embedded. This setting is valid only when `CompatibilityLevel` is PDF 1.5 or later.

Embedding means that the multimedia object is included in the PDF file itself. Linking means that the PDF file contains a reference to an external file. (The user should make sure that the media files are in the same directory as the PDF file.)

Supported by

InDesign

Type

name

UI name

Multimedia

Default value

`UseObjectSettings`

PageMarksFile

Specifies selection of the page marks file. Valid values are:

`RomanDefault`: Use the built-in Roman page marks file

`JapaneseWithCircle`: Use the built-in Japanese (with circle) page marks file.

`JapaneseNoCircle`: Use the built-in Japanese (with no circle) page marks file.

`UseName`: Use the value of `PageMarksFileName` for the page marks file

The files presented in the user interface vary according to application and language.

Supported by

Illustrator, InDesign

Type

name

UI name

InDesign: "Type" pop-up

Illustrator: Printer Mark Type

Default value

`RomanDefault`

PageMarksFileName

If `PageMarksFile` has the value `UseName`, this setting represents the page marks file that should be used.

Supported by

Illustrator, InDesign

Type

string

UI name

Page mark file name

Default value

" "

PDFXOutputIntentProfileSelector

Specifies the PDF/X output intent profile to be used for PDF/X compliance. The values are similar to those for the `DestinationProfileSelector` setting.

Valid values are:

Value	UI equivalent	Description
`NA`	N/A	Indicates that no PDF/X compliance standard has been specified.
`UseName`	*Any named profile*	The profile specified by `PDFXOutputIntentProfile` is used as the PDF/X output intent profile.
`WorkingCMYK`	Working CMYK	The effective working CMYK profile is used as the PDF/X output intent profile.
`WorkingRGB`	Working RGB	The effective working RGB profile is used as the PDF/X output intent profile.
`DocumentCMYK`	Document CMYK	The effective document CMYK profile is used as the PDF/X output intent profile.
`DocumentRGB`	Document RGB	The effective document RGB profile is used as the PDF/X output intent profile.

Note: When color management is on and PDF/X compliance has been specified, the effective profiles specified by `DestinationProfileName`, `CalCMYKProfile` and `PDFXOutputIntentProfile` must be the same.

See "Using the PDF/X output intent settings" on page 44 for more information about how this setting is used along with the other standards settings.

Supported by

Illustrator, InDesign, Photoshop

Type

name

UI name

Output Intent Profile Name

Default value

`DocumentCMYK`

PreserveEditing

If `true`, data is saved in the PDF that allows you to subsequently re-open and edit the data in the authoring application.

Note: This capability is not available when specifying PDF/X compliance (see `CheckCompliance` on page 125).

Supported by

Illustrator, Photoshop

Type

Boolean

UI name

Preserve *application name* Editing Capabilities

Default value

`true`

UntaggedCMYKHandling

Specifies whether untagged CMYK content (content with no embedded profiles) should be tagged. Valid values are:

`LeaveUntagged`: Do not add profiles to untagged objects.

`UseDocumentProfile`: Use document CMYK profile for untagged objects.

Note: In InDesign and Illustrator, this value applies to placed content originating in other applications. Native content is treated as inherently untagged unless the option to include all profiles is chosen.

See "Using the color conversion settings" on page 34 for details of how these settings interact with the other color settings and how they appear in the UI.

Supported by

Illustrator, InDesign, Photoshop

Type

name

UI name

Output panel: Color

Default value

`LeaveUntagged`

UntaggedRGBHandling

Specifies whether untagged RGB content (content with no embedded profiles) should be tagged. Valid values are:

`LeaveUntagged`: Do not add profiles to untagged objects.

`UseDocumentProfile`: Use document RGB profile for untagged objects.

See "Using the color conversion settings" on page 34 for details of how these settings interact with the other color settings and how they appear in the UI.

Supported by

all applications

Type

name

UI name

Output panel: Color

Default value

`UseDocumentProfile`

UseDocumentBleed

If `true`, the bleed values (otherwise specified by `BleedOffset`) are populated based on the document's bleed.

Supported by

InDesign

Type

Boolean

UI name

Use Document Bleed

Default value

`false`

InDesign namespace settings

The following settings belong to the `InDesign` namespace. They are supported by InDesign only.

AsReaderSpreads

If `true`, spreads are exported as single PDF pages. A *spread* is a set of pages that are viewed together, such as the two pages that are visible when a magazine is opened.

Type

Boolean

UI name

Spreads

Default value

`false`

CropImagesToFrames

If `true`, only data that represents the visible (non-cropped) part of an image is exported in the PDF file.

Note: InDesign crops images to the visible portion of the frame. Distiller has settings for cropping (see "Disabling of image cropping" on page 33) that use the clip area.

Type

Boolean

UI name

Crop Image Data to Frames

Default value

`true`

ErrorControl

Specifies what should be done when errors occur when generating a PDF file. The following table shows the valid values.

Value	Description
`WarnAndContinue`	A warning dialog box is issued. The user can choose to continue and the PDF file will be produced. If the UI is disabled in scripting, the dialog box is automatically dismissed.
`Ignore`	Errors are ignored.
`CancelJob`	No PDF file is produced.

Type

name

UI name

none (supported via scripting)

Default value

`WarnAndContinue`

FlattenerIgnoreSpreadOverrides

If `true`, transparency flattener settings are applied to all spreads in a document, overriding the flattener settings for individual spreads.

Type

Boolean

UI name

Ignore Spread Overrides

Default value

`false`

IncludeGuidesGrids

If `true`, visible grids and guides will appear in the PDF document.

Type

Boolean

UI name

Visible Guides and Baseline Grids

Default value

`false`

IncludeNonPrinting

If `true`, non-printing objects (which have had the Nonprinting option applied in the Attributes palette) will appear in the PDF document.

Type

Boolean

UI name

Non-Printing Objects

Default value

`false`

IncludeSlug

If `true`, the slug area (an area outside the page and bleed that contains information such as printer instructions) is included in the PDF document.

Type

Boolean

UI name

Include Slug area

Default value

`false`

OmitPlacedBitmaps

If `true`, imported bitmap images are not included in the PDF document but are replaced by OPI comments.

Type

Boolean

UI name

Omit for OPI: bitmap images

Default value

`false`

OmitPlacedEPS

If `true`, imported Encapsulated PostScript (EPS) images are not included in the PDF document but are replaced by OPI comments.

Type

Boolean

UI name

Omit for OPI: EPS

Default value

`false`

OmitPlacedPDF

If `true`, PDF that has been placed (imported) into the document is not included in the PDF document but are replaced by OPI comments.

Type

Boolean

UI name

Omit for OPI: PDF

Default value

`false`

SimulateOverprint

Simulates the appearance of printing separations by maintaining the appearance of overprinting in composite output. The following values are valid.

Value	UI equivalent	Description
`Legacy`	unchecked	Preserve overprint & spot colors when flattening.
`SimulatePress`	checked	Spot colors are changed to process equivalents. The transparency flattener simulates the appearance of overprinted objects using process colors.

This setting applies only when the following conditions are met:

- `CompatibilityLevel` is 1.3 (Acrobat 4) or earlier (that is, when transparency is not supported)
- There is a CMYK or RGB output color space.

Type

name

UI name

Simulate Overprint check box

Default value

`Legacy`

5 Conversions Related to JDF

This appendix describes how Distiller converts `setpagedevice`-type PostScript key-word pairs and parameters into a job description file (JDF). Distiller creates a JDF file if the `CreateJDFFile` parameter is set to true. The version of JDF created is 1.1. For more information, see the JDF specification at http://www.cip4.org/documents/jdf_specifications.

This appendix uses XPath expressions to identify specific attributes. *XPath* is a language for addressing parts of an XML document, as defined in XML Path Language (XPath) Version 1.0, which is available from http://www.w3.org/TR/xpath. The conventions that appear in the following tables are shown below:

Expression ::= JDFRoot'/'Attribute | JDFRoot'/'Children'/'Attribute

JDFRoot ::= '/JDF'

Children ::= Element | Element'/'Children

Element ::= element

Attribute ::= '@'attribute

Creation of the basic JDF file

Distiller creates a basic JDF document whose root node is a JDF element with Type="Product". Under that root node, Distiller creates three sub-elements:

- A JDF element with `Type = "Combined"`
- A ResourcePool element that describes the document produced
- An AuditPool element that describes the results of distillation

The resulting root node is populated with elements that describe the incoming PostScript stream, `Combined` process node, and the following items:

- `setpagedevice`-*type operators*. Whenever Distiller encounters a supported `setpagedevice`-type operator, it represents the key value as an entry in one of the parameter resources associated with the `ResourceDedfinition` process. (See "Representation of PostScript keys as JDF entries" on page 15.)
- *DSC comments*. Whenever Distiller encounters certain DSC comments, it represents those comments in the RunList for the PDF file. (See "Mapping of DSC comments into JDF elements and attributes" on page 18.)
- *Parameters*. Distiller creates a `PSToPDFConversionParams` resource which it populates with attributes that correspond to the settings of the parameters as of the end of the first page of the job. If the parameter `ColorConversionStrategy` is NOT `LeaveColorUnchanged`, Distiller also creates a `ColorSpaceConversionParams` resource, which it populates as it does for `PSToPDFConversionParams`. (See "Mapping of parameters into JDF elements and attributes" on page 19.)

Representation of PostScript keys as JDF entries

Distiller represents selected `setpagedevice`, `settrapparams`, or `settrapzone` PostScript key-word pairs as JDF entries. It does so by creating and populating corresponding JDF resource elements in a

`ResourceDefinition` resource pool, as described in the table PostScript keys converted into JDF ResourceDefinition resources.

On occasion, a PostScript key contradicts a Distiller parameter. For information on how this conflict is resolved, see *Using Adobe Normalizer Server, Version 6.0.4*, section 7.2.

PostScript keys converted into JDF ResourceDefinition resources

PostScript key	Representation in JDF ResourceDefinition resources
`setpagedevice`	Distiller converts the `setpagedevice` key-word pairs into the `ResourceDefinition` attributes described in the table "Mapping from setpagedevice keys to JDF entries" on page 15. Some keys are omitted from the table because they do not have logical equivalents in the *JDF Specification*.
`settrapparms`	Distiller converts the `settrapparam` key-word pairs into the ResourceDefinition attributes described in the table "Mapping from settrapparms keys to JDF TrappingDetails entries" on page 16.
`settrapzone`	Distiller converts the `settrapzone` key-word pairs into the ResourceDefinition attributes described in the table "Mapping from settrapzone keys to JDF TrappingDetails entries" on page 17.

Mapping from setpagedevice keys to JDF entries

Key Name	Entry in /JDF / ResourcePool
`Collate`	*`DigitalPrintingParams /@ Collate`*
`DeviceRenderingInfo <<ValuesPerColorComponent>>`	*`RenderingParams /@ ColorantDepth`*
`Duplex`	*`LayoutPreparationParams /@ Sides`*
`HWResolution`	*`RenderingParams / ObjectResolution /@ Resolution`*
`Jog`	*`Component / Disjointing /@ OffsetAmount`*
`ManualFeed`	`DigitalPrintingParams /@ ManualFeed`
`MediaColor`	`DigitalPrintingParams / Media /@ MediaColorName`
`MediaPosition`	`DigitalPrintingParams / Media / Location /@ LocationName`
`MediaType`	`DigitalPrintingParams / Media /@ UserMediaType`
`MediaWeight`	`DigitalPrintingParams / Media /@ Weight`
`MirrorPrint`	`ImageSetterParams /@ MirrorAround`
`NegativePrint`	`ImageSetterParams /@ Polarity`
`NumCopies`	If `Collate` is `TRUE`, `RunList /@ PageCopies`. Otherwise, `RunList /@ DocCopies`

Key Name	Entry in /JDF / ResourcePool
PageSize	DigitalPrintingParams / Media /@ Dimension
ProcessColorModel	ColorantControl /@ ProcessColorModel
SeparationColorNames	ColorantControl /@ ColorantParams
SeparationOrder	ColorantControl /@ ColorantOrder
Separations	ColorantControl /@ ForceSeparations
Trapping	TrappingDetails /@ Trapping
TrappingDetails <<Type>>	TrappingDetails /@ TrappingType
TrappingDetails <<ColorantDetails <<ColorantName>> >>	ColorantControl / ColorPool /@ ColorName
TrappingDetails <<ColorantDetails <<ColorantType>> >>	ColorantControl / ColorPool / Color /@ ColorType
TrappingDetails <<ColorantDetails <<NeutralDensity>> >>	ColorantControl / ColorPool / Color /@ NeutralDensity
TrappingDetails <<TrappingOrder>>	TrappingDetails /@ TrappingOrder
Tumble	LayoutPreparation /@ Sides

Mapping from settrapparms keys to JDF TrappingDetails entries

Key Name	Entry in /JDF / ResourcePool / TrappingDetails / TrapRegion / TrappingParams
BlackColorLimit	@ BlackColorLimit
BlackDensityLimit	@ BlackDensityLimit
BlackWidth	@ BlackWidth
ColorantZoneDetails <<StepLimit>>	@ ColorantZoneDetails /@ StepLimit
ColorantZoneDetails <<TrapColorScaling>>	@ ColorantZoneDetails /@ TrapColorScaling
ColorantZoneDetails <<TrapPlacement>>	@ ColorantZoneDetails /@ ADBE[a]:TrapPlacement
Enabled	@ Enabled
ImageInternalTrapping	@ ImageInternalTrapping

Key Name	Entry in /JDF / ResourcePool / TrappingDetails / TrapRegion / TrappingParams
ImageMaskTrapping	@ ImageMaskTrapping
ImageResolution	@ ImageResolution
ImageToImageTrapping	@ ImageToImageTrapping
ImageToObjectTrapping	@ ImageToObjectTrapping
ImageTrapPlacement	@ ImageTrapPlacement
ImageTrapWidth	@ ADBE:ImageTrapWidth
MinimumBlackWidth	@ MinimumBlackWidth
SlidingTrapLimit	@ SlidingTrapLimit
StepLimit	@ StepLimit
TrapColorScaling	@ TrapColorScaling
TrapEndStyle	@ TrapEndStyle
TrapJoinStyle	@ TrapJoinStyle
TrapWidth	@ TrapWidth

a. In the JDF document, Distiller defines ADBE as the namespace ***http://ns.adobe.com/JDF***.

Mapping from settrapzone keys to JDF TrappingDetails entries

Key Name	Entry in /JDF / ResourcePool / TrappingDetails
settrapzone	TrapRegion /@ TrappingZone

Conversion of the linear representation of setpagedevice keys

The `setpagedevice` keys that appear in PostScript streams/files are presented in a linear fashion. That is, hierarchical relationships are represented by repeating higher level information. In contrast, the JDF format represents a hierarchy.

This section describes how Distiller converts between that linear representation and the JDF hierarchy.

For example, the *JDF Specification* allows the `TrapParams` resource element to appear as a child of the `TrappingDetails` resource element and the `TrapZones` resource element, as in the following code.

```
<TrappingDetails>
   <TrapRegion>
      <TrapParams …/>
   </TrapRegion.>
</TrappingDetails>
```

Distiller always subordinates `TrapParams` resources to `TrapRegion` resources. That is, Distiller never produces entries, such as the first `TrapParams` resource.

Instead, if Distiller has set a default trapping zone, it is set for all the pages (using the second TrapParam). Subsequently any settrapzone/settrapparam settings cause a new TrapZone with associated TrapParams. There can be many of these per page.

`TrapRegions` elements (with associated `TrapParams` elements) are created from each `settrapzone` PostScript call using the `trapparams` set at the time (by `settrapparams`) and the `Page` key is set. Default `trapzones` (set as part of an unencapsulated PostScript job as per the *PostScript Language Reference*) are turned in to a `trapregion` that applies to all pages.

More specifically, the trapping settings may be different for two separate regions of a particular page. For example, the title text and logo of a page might have different settings compared to those used for the body text. A particular image could then also have different settings. As a result, a `TrapZone` is drawn around each object (a normal PostScript path) and different `trapparams` set for each object.

Mapping of DSC comments into JDF elements and attributes

The presence of the `%%Page` DSC comment in a PostScript stream indicates the beginning of a page in the stream. The presence of the %%PlateColor DSC comment in conjunction with `%%Page` indicates the beginning of a pre-separated page for a particular colorant.

Distiller may use the `%%Page` and `%%PlateColor` comments to create a partitioned `RunList` that represents the structure of the full-document PDF file it produces for a PostScript stream, depending on the document structure implied by those comments, as described in the following subsections. The `RunList` is in the ResourcePool, which is at the same level as the Combined process node.

Composite jobs

Composite jobs are indicated by the absence of any `%%PlateColor` comments in the PostScript stream.

Normalizer produces un-partitioned `RunLists` for composite jobs. Changes in page device key values are not considered.

Pre-separated jobs with interleaved separations

Pre-separated jobs with interleaved separations are indicated by the appearance of `%%PlateColor` comments soon after each `%%Page` comment, with each `%%PlateColor` specifying the next colorant in the sequence. That is, the separations that compose a single page appear sequentially, (i.e. cyan separation, magenta separation, yellow separation, and black separation).

When processing pre-separated jobs with interleaved separations, Distiller uses the `%%Page` and `%%PlateColor` DSC comments to create a `RunList` element partitioned on the keys `Run` and `Separation` and to create a `RunIndex` that references the pages in that `RunList`.

Distiller creates an additional `RunIndex` range for the pages that apply to each set of page device key values.

Pre-separated single-colorant jobs

Pre-separated single-colorant jobs are the same as Pre-separated jobs with interleaved separations, except all `%%PlateColor` comments describe a single colorant.

When processing pre-separated single-colorant jobs, Distiller uses the `%%Page` and `%%PlateColor` DSC comments as described for pre-separated jobs with interleaved separations, except the RunList contains a single partition with `Run=1` and `Separation` set to the colorant name provided in `%%PlateColor`.

Mapping of parameters into JDF elements and attributes

This section describes how Distiller converts parameter settings into JDF element and attribute settings. It presents one section for each category of parameter, as follows:

- General
- Image compression
- Fonts
- Color conversion
- Advanced

Distiller produces only those JDF attributes described in this section. Some parameters (such as `CreateJobTicket`) do not have JDF attribute counterparts. In contrast, some JDF attributes in applicable elements do not correlate with parameters, such as the `RenderingIntent` attribute in the `ColorSpaceConversionParams` element.

Distiller represents parameters as values for the attributes in the following resource elements:

- `PSToPDFConversionParams`
- `FontParams`
- `ImageCompressionParams`
- `ColorSpaceConversionParams`

Distiller does not create the optional `ColorantControl` element.

Note: Version 1.1a of the JDF Specification changed the `ColorantControl` element in a PSToPDFConversion process node from required to optional.

General

The following table specifies the conversion from Distiller general parameters into JDF elements.

Conversion from Distiller general parameters into JDF attributes

Parameter	Attribute name in the PSToPDFConversionParams resource
AutoRotatePages	@ AutoRotatePages
Binding	@ Binding
CompatibilityLevel	@ PDFVersion and ColorSpaceConversionParams /@ Operation The table "Conversion from ColorConversionStrategy into Operation" on page 23 describes the role of CompatibilityLevel in deriving the Operation value
CompressObjects	@ADBE:CompressObjects
CoreDistVersion	Not represented in JDF. CoreDistVersion is a read-only parameter that is meaningless in JDF.
DoThumbnails	@ DoThumbnails
EndPage	@ EndPage
ImageMemory	@ ImageMemory
Optimize	@ Optimize
StartPage	@ StartPage

Image compression

The Distiller image compression parameters map into the JDF ImageCompressionParams element, which may have up to three ImageCompression subelements, one for each of the following image types:

- Color
- Grayscale
- Monochrome

Each ImageCompression subelement contains an ImageType attribute that identifies the type of image it represents.

Conversion from Distiller Image Compression parameters into JDF ImageCompression subelement

Parameter	Attribute name in the color, grayscale or monochrome ImageCompression subelement
`AntiAliasColorImages`, `AntiAliasGrayImages`, or `AntiAliasMonoImages`	`@ AntiAliasImages`
`AutoFilterColorImages` or `AutoFilterGrayImages` (Not relevant to monochrome images.)	`@ AutoFilterImages`

`AutoFilterColorImages`
`AutoFilterGrayImages`
`/ColorACSImageDict <</QFactor>> /GrayACSImageDict <</QFactor>> /ColorImageDict <</QFactor>> /GrayImageDict <</QFactor>> /MonoImageDict <</QFactor>>`
The table "Conversion from Parameters into the JDF DCTQuality attribute" on page 22 describes how the auto filter parameter influences selection of a /QFactor value.
The compression quality dictionaries described above may contain other factors that influence compression; however, they are not represented in JDF attributes.
	`@ DCTQuality`
	Distiller calculates `DCTQuality` by dividing the selected QFactor by 100. For example:
	QFactor / **DCTQuality**
	2.40 / 0.0240
	1.30 / 0.0130
	0.15 / 0.0015

`ColorImageDepth`, `GrayImageDepth`, or `/MonoImageDepth`	`@ ImageDepth`
`ColorImageDownsampleThreshold`, `GrayImageDownsampleThreshold`, or `MonoImageDownsampleThreshold`	`@ ImageDownsampleThreshold`
`ColorImageDownsampleType`, `GrayImageDownsampleType`, or `MonoImageDownsampleType`	`@ ImageDownsampleType`
`ColorImageFilter`, `GrayImageFilter`, or `MonoImageFilter`	`ImageFilter` or `ADBE:ImageFilter` The latter being used to represent the value `JPXEncode`, `LZWEncode`, or `RunLengthEncode`.
`ColorImageResolution`, `GrayImageResolution`, or `MonoImageResolution`	`@ ImageResolution`

Parameter	Attribute name in the color, grayscale or monochrome ImageCompression subelement

`JPEG2000ColorACSImageDict <</Quality>>`
`JPEG2000GrayACSImageDict <</Quality>>`
`JPEG2000ColorImageDict <</Quality>>`
`JPEG2000GrayImageDict <</Quality>>`
	`@ADBE`[a]`:JPXQuality`

`ConvertImagesToIndexed`	`@ ConvertImagesToIndexed` (Represented only in an `ImageCompressionParams` element with `ImageType = "Color"`.)
`DownsampleColorImages`, `DownsampleGrayImages`, or `DownsampleMonoImages`	`@ DownsampleImages`
`EncodeColorImages`, `EncodeGrayImages`, or `EncodeMonoImages`	`@ EncodeImages`

a. ADBE must be defined as the namespace http://ns.adobe.com/JDF. That is, JDF files that contain elements or attributes that use the ADBE prefix must also contain the definition xmlns:ADBE="http://ns.adobe.com/JDF".

Conversion from Parameters into the JDF DCTQuality attribute

Condition	Distiller compression dictionary key-word pair used to derive the value of *ImageCompression /@ DCTQuality*
If AutoFilterColorImages is `true`	`/ColorACSImageDict <</QFactor>>`
If AutoFilterColorImages is `false`	`/ColorImageDict <</QFactor>>`
If AutoFilterGrayImages is `true`	`/GrayACSImageDict <</QFactor>>`
If AutoFilterGrayImages is `false`	`/GrayImageDict <</QFactor>>`
If AutoFilterGrayImages value is irrelevant	`/MonoImageDict <</QFactor>>`

Page compression

`CompressPages` is the sole Distiller page compression parameter. Distiller converts it into the `PSToPDFConversionParams CompressPages` attribute.

Fonts

Distiller converts each Distiller font parameter into the attribute in the JDF `FontParams` resource element with the same name. In other words, for each Distiller font parameter, there is an identically-named attribute in the `FontParams` element.

Color conversion

If `ColorConversionStrategy` is `LeaveColorUnchanged`, `ColorSpaceConversionParams` element is omitted from the JDF. Otherwise, conversion is as described in the following table.

Conversion from Distiller color conversion parameters to JDF ColorSpaceConversionParams attributes

Parameter	Attribute name in ColorSpaceConversionParams
`CalCMYKProfile` Used as the ICC profile FileSpec in the `ColorSpaceConversionOp` resource that contains Type = "CMYK".	`FileSpec` and `@ Type`
`CalGrayProfile` Used as the ICC profile FileSpec in the `ColorSpaceConversionOp` resource that contains Type = "Gray".	`ColorSpaceConversionOp / FileSpec` and `ColorSpaceConversionOp /@ Type`
`CalRGBProfile` Used as the ICC profile FileSpec in the `ColorSpaceConversionOp` resource that contains Type = "RGB".	`ColorSpaceConversionOp / FileSpec` and `ColorSpaceConversionOp /@ Type`
`ColorConversionStrategy`	`ColorSpaceConversionOp /@ Operation` and `ColorSpaceConversionOp /@ SourceObjects`, as described in the tables "Conversion from ColorConversionStrategy into Operation" on page 23 and "Conversion from /ColorConversionStrategy into SourceObjects" on page 24.
`DefaultRenderingIntent`	`PSToPDFConversionParams /@ DefaultRenderingIntent`
`PreserveHalftoneInfo`	`@ PreserveHalftoneInfo`
`PreserveOverprintSettings`	`@ PreserveOverprintSetting`
`sRGBProfile`	`FileSpec` If ColorConversionStrategy is sRGB, Distiller creates a FileSpec element with Usage="FinalTargetDevice" and a UID value that reflects the ICC profile used for converting color spaces to CalRGB (PDF 1.2) or sRGB (PDF 1.3 and above).
`TransferFunctionInfo`	`@ TransferFunctionInfo`
`UCRandBGInfo`	`@ UCRandBG`
None; however, Distiller specifies the built-in color management system.	`@ ColorManagementSystem`

Conversion from ColorConversionStrategy into Operation

ColorConversionStrategy value	Operation attribute value
Device independent color[a] and CompatibilityLevel <= 1.2	`Convert`
Device independent color and CompatibilityLevel > 1.2	`Tag`
sRGB	`Convert`

a. ColorConversionStrategy == UseDeviceIndependentColor || ColorConversionStrategy == UseDeviceIndependentColorForImages.

Conversion from /ColorConversionStrategy into SourceObjects

ColorConversionStrategy value	SourceObjects attribute value
`UseDeviceIndependentColor`	`All`
`sRGB`	`All` and `FinalTargetDevice` set to sRGB. **Note:** If the conversion is sRGB, then we do NOT create a `ColorSpaceConversionOp` of `SourceCS = Gray` because the Gray colors are not changed.
`UseDeviceIndependentColorForImages`	`ImagePhotographic` `ImageScreenShot`

Advanced

The following table specifies the conversion from Distiller advanced parameters into JDF elements.

Conversion from Distiller advanced parameters into JDF elements

Parameter	Attribute name in the PSToPDFConversion resource
`AllowPSXObjects`	`@ADBE`[a]`:AllowPSXObjects`
`AllowTransparency`	`@ADBE:AllowTransparency`
`ASCII85EncodePages`	`@ ASCII85EncodePages`
`AutoPositionEPSFiles`	`AdvancedParams /@ AutoPostitionEPSInfo`
`CreateJobTicket`	Not represented in JDF.
`DetectBlends`	`@ DetectBlend` **Note:** This is the correct spelling.
`EmbedJobOptions`	`@ADBE:EmbedJobOptions`
`EmitDSCWarnings`	`AdvancedParams /@ EmitDSCWarnings`
`LockDistillerParams`	`AdvancedParams /@ LockDistillerParams`
`OPM`	`@ OverPrintMode`

Parameter	Attribute name in the PSToPDFConversion resource
ParseDSCComments	AdvancedParams /@ ParseDSCComments
ParseDSCCommentsForDocInfo	AdvancedParams /@ ParseDSCCommentsForDocInfo
PassThroughJPEGImages	@ADBE:PassThroughJPEGImages
PreserveCopyPage	AdvancedParams /@ PreserveCopyPage
PreserveEPSInfo	AdvancedParams /@ PreserveEPSInfo
PreserveOPIComments	AdvancedParams /@ PreserveOPIComments
UsePrologue	AdvancedParams /@ UsePrologue

a. In the JDF document, Distiller defines ADBE as the namespace http://ns.adobe.com/JDF .

PDF/X

The following table specifies the conversion from Distiller PDF/X parameters into JDF elements.

Conversion from Distiller PDF/X parameters into JDF elements

Parameter	ADBE:PDFXParams attribute name
PDFX1aCheck	@ADBE : PDFX1aCheck
PDFX3Check	@ADBE : PDFX3Check
PDFXCompliantPDFOnly	@ADBE : PDFXCompliantPDFOnly
PDFXNoTrimBoxError	@ADBE : PDFXNoTrimBoxError
PDFXTrimBoxToMediaBoxOffset	PDFXTrimBoxToMediaBoxOffset
PDFXSetBleedBoxToMediaBox	PDFXSetBleedBoxToMediaBox
PDFXBleedBoxToTrimBoxOffset	PDFXBleedBoxToTrimBoxOffset
PDFXOutputIntentProfile	PDFXOutputIntentProfile
PDFXOutputCondition	PDFXOutputCondition
PDFXRegistryName	PDFXRegistryName
PDFXTrapped	PDFXTrapped

Conversion of parameters not available through the user interface

All parameters that cannot be set through the user interface are converted into attributes in the ADBE:ThinPDFParams element, as specified in the following table.

Conversion from parameters that cannot be set through the Distiller UI

Parameter set using the setdistillerparam key	ADBE:ThinPDFParams attribute name
`sidelineEPS`	`@ ADBE`[a]`:SidelineEPS`
`filePerPage`	`@ FilePerPage`
`sidelineFonts`	`@ SidelineFonts`
`sidelineImages`	`@ SidelineImages`

a. In the JDF document, Distiller defines ADBE as the namespace http://ns.adobe.com/JDF.

Index

A

B

C

D

E

F

G

H

I

J

L

M