How do I save the smallest PDF files possible?

Compressing PDF documents

Datalogics provides advanced compression ability for PDF documents with several of our products:

- The Adobe PDF Library PDF Optimizer API, supported by all three interfaces, Java, .Net, and C++
- Our PDF Optimizer command line application, available in Windows and Linux, distributed with our free PDF Checker utility
- On the Cloud with the Datalogics PDF REST APIs

With PDF Optimizer and with our PDF REST APIs, we provide three compression options,

- **compressionLow**. Best suited for users who need to make their PDF documents smaller but who don't want to compromise those documents.
- **compressionMedium**. Substantially compress PDF documents while managing the results in the output files. This would be a standard choice, a compromise between shrinking the size of a PDF document as much as possible while limiting the impact on that output file.
- **compressionHigh**. Choose maximum possible compression regardless of the impact on the resulting PDF output files.

The more you compress a PDF document, the greater the likely impact on that document. For example, if you really want to compress a PDF document as small as possible, to make it fast and easy to distribute, your clients and customers might notice that the quality of images in that PDF document have been reduced, especially if the document is printed. Also, one way to reduce the size of a PDF document is to remove fonts that are saved within the PDF document itself. That, however, tends to make the file less portable. If a font is not embedded within the PDF document, the viewer used to read that PDF document must find the necessary fonts installed on the local computer hardware instead. If the viewer can't find the font it needs, it will look for a reasonable substitute. That often works—Times New Roman is likely to look pretty much the same regardless of the font typeset used. But sometimes a PDF viewer can substitute a font and end up with a PDF file that looks different. And if the viewer needs a local Mandarin or Arabic font and can't find it installed on the machine, the PDF file that appears in the viewer would be unreadable.

Note that beyond these three compression settings for the PDF Optimizer API and for the PDF REST APIs, we also provide a means for you to enter custom compression settings, giving you detailed control over how the software compresses your PDF documents.

You can experiment with Datalogics PDF compression by using our free Compression Web Service.

Working with flags in the PDDocSaveParams element in the Adobe PDF Library

To reduce the size of PDF documents when saving them, you can use severa flags within the PDDocSaveParams element, as part of elements saveFlags and saveFlags2.These flags are passed to the PDDocSaveWithParams() API call.

See the discussion of the PDF Optimizer, related to the Adobe PDF Library. The same utility is provided for both the Core Library and for the Java and Microsoft .NET Interfaces.

saveFlags:

- **PDSaveFull**: Causes the PDF file to be fully written. Incrementally saved versions of the PDF document are discarded and the PDF to only contains the most recent contents. Do not use if you want to preserve previous revisions of the PDF file.
- **PDSaveCollectGarbage**: Causes objects in the PDF that are unreferenced to be discarded.

saveFlags2:

- **PDSaveCompressed**: Allows for PDF object compression and collections to be used. When this flag is set, the PDF can contain sets of compressed objects that are unpacked into their respective PDF objects when the PDF document is opened.
- **PDSaveRemoveASCIIFilters**: Allows for ASCII85 filters to be removed from the PDF document. This change can allow for some streams to contain binary (non-ASCII) data, reducing the file size. But with this flag set the PDF document will not be able to be transmitted through channels that do not properly handle binary data. This is unlikely to be a problem; the common FTP and HTTP transfer protocols will work fine. Protocols that have problems with binary data are obscure and rarely used any longer.
- **PDSaveAddFlate**: Compresses PDF streams with no compression using the Flate algorithm, except for metadata streams, which are left uncompresed.
- **PDSaveReplaceLZW**: Causes LZW-compressed PDF streams to be re-compressed with Flate. This can save some space in most cases.
- **PDSaveOptimizeXObjects**: Causes the Adobe PDF Library to search for identical forms and images and to reconcile these to references to a single form or image instead of references to the multiple, identical forms or images. This can involve substantial overhead during saving.

Other flags to consider.

saveFlags:

- **PDSaveLinearized**: This flag optimizes the PDF for fast web viewing and random accesses of pages. Linearizing a PDF document places heavy demands on the local workstation's memory and processor capacity, however, because all of the PDF objects in the saved file must be examined and potentially reordered. Use this flag with your application only if creating Linearized PDF documents is important enough that you can live with the memory and processor requirements.

    > A linearized PDF document is processed in a way that allows for more efficient display of long PDF documents. A linearized PDF is restructured in a way that allows the first page of the file to appear on a user's web browser while the rest of the file is being downloaded. This type of PDF document can thus display more quickly on a web page; the user does not need to wait for the entire document appear before he or she can start reading it. This is also known as web optimization.

saveFlags2:

- **PDSaveOptimizeContentStreams**: Attempts to consolidate common preambles between content streams when saving PDF files. These are not found very often.

- **PDSaveOptimizeFonts**: Consolidates duplicated fonts and font subsets into references to one unified font. This can involve substantial memory and CPU overhead, however. It is also appropriate for PDF files created from merging many different PDF files into a single document, or PDF files created by merging pages from several different PDF files.
- **PDSaveOptimizeMarkedJBIG2Dictionaries**: Consolidate out unused symbols from JBIG2-encoded datastreams in the PDF file. This is rarely used.