Datalogics

Datalogics PDF Checker™

User Guide

PDF Checker version 2.5

For additional information, contact:

Datalogics, Incorporated
1207 Delaware Ave.
Suite # 1810
Wilmington, DE 19806
Phone: (312) 853-8200
www.datalogics.com

Table of Contents

Introduction

PDF technology today

More than 25 years after the Portable Document Format was introduced, PDF documents continue to dominate the digital marketplace. Some 2.5 trillion PDF files are produced each year. PDF documents are widely used to distribute financial statements, reports, articles, proposals, and other long documents that feature complex formatting. They appear all over the Internet and are extensively used by businesses and government agencies, frequently to replace printed documents for conveying and storing content. PDFs can be opened in any browser window, and their formatting is perfectly preserved when sent to a printer. PDF documents are also easy to secure, they can be used as electronic forms, and you can add digital signatures to PDF files, thus turning them into legal documents.

But this very popularity creates a problem in managing PDF documents. PDF files can be generated using a dizzying array of software systems and platforms, including many that are out of date, as it is also common to find PDF files that are five, ten, even 15 years old. That means that it is sometimes hard to tell what is stored inside any given PDF file, and how those features—attached files, graphics, form fields, signatures, and so on—might affect how you can manage that PDF document or distribute it. Can you make a massive PDF document smaller, somehow, without compromising its quality? Can unnecessary features be removed so that the document will print faster or open faster in a browser window? What about fixing errors within the PDF itself?

Using PDF Checker

That's what PDF Checker is for. PDF Checker is a free, simple scriptable server tool for 64-bit Windows and Linux platforms that allows you to quickly scan a PDF document or set of documents to look for problems, or to simply identify features within a document that are likely to get in your way if you want to use the PDF efficiently.

Then, with this knowledge of your PDF documents, you can use another Datalogics product, PDF Optimizer, to apply fixes and improvements to those documents so that they will download faster or open more quickly in a browser window or mobile device.

You can enter a PDF Checker statement from a command prompt, and manually check one PDF document at a time. Or you could add a PDF Checker command statement to a batch file. With a batch file, you can create a workflow that uses PDF Checker to check large numbers of PDF documents automatically. In this case you might need to write some JavaScript or Python code to automatically generate a series of command line statements, each with a unique name for the input PDF document, JSON profile, output report file, and if necessary, the password to open a PDF input document.

We provide detailed online documentation about PDF Checker and our other products at our Developer Resources site. Visit https://docs.datalogics.com/PDFChecker.

Note that the Windows installation program for PDF Checker adds the location of the PDF Checker executable to the PATH in the Windows Environment Variables, so you can run "pdfchecker.exe" from anywhere. For Linux, if you want to run the executable from anywhere, you need to manually add the location of the PDF Checker executable to the PATH variable.

Installing PDF Optimizer with PDF Checker

PDF Checker can be used as an independent tool to assess the characteristics of PDF documents, but it was also designed to be used with PDF Optimizer to conditionally apply appropriate fixes and improvements. For this reason, we provide a free trial version of PDF Optimizer with your PDF Checker installation.

You will receive an activation key for your 14 day trial of PDF Optimizer via the email sent to the address that you provided. The PDF Optimizer user guide, located in the program's installation directory, provides further activation instructions. You can also find online documentation for PDF Optimizer at our Developer site, https://docs.datalogics.com/PDFOptimizer.

Using the JSON profile

The JSON profile is a file that defines search conditions for PDF Checker to use when scanning an input PDF document. JSON, or JavaScript Object Notation, is an open standard file format that relies on easily readable English text, and it is used as an alternative to XML. The custom settings that you provide to PDF Checker to manage the process are all defined in this profile. The name of the profile file must be included in the command line statement.

You could, for example, edit your JSON profile so that PDF Checker looks for unembedded fonts and flags any it finds in a report as an error or information statement. Then the software generates a report listing the results. You can display the results as standard output from your command line tool, and either list the results on the screen or use a program or script to redirect the standard output to an external file. You can also use an option in the command that directs the PDF Checker results to a text file that you define.

We provide a default profile file, everything.json, with the product that includes every available search option. Feel free to edit this file or use it as a model for creating your own. You can name your profile file or files whatever you like and store them wherever you want to. But the content of your profile must be valid JSON content, and we recommend that when you name a profile file you include the ".json" file extension.

If you want to create your own custom profile, we recommend that you save a copy of the default JSON file and rename it, and then edit this copy. This way you will preserve the original JSON profile for later reference. Also, if you install an updated version of the software, the installation process will overwrite the original file, and any changes you made to that file will be lost. Besides saving your own copy of the profile, you might also want to create a backup your edited JSON profile in a different directory.

You can use the JSON validator JSONLint to check your JSON syntax (https://jsonlint.com).

Command Syntax

The command syntax for PDF Checker includes these values:

- **Required:** The executable name, `pdfchecker`
- **Required:** `-i` [`--input`] name and path of the PDF input file to review
- **Required:** `-j` [`--profile`] name and path of the JSON profile file
- **Optional:** `-o` [`--output`] name and path to assign to the output results file
- **Optional:** `-s` [`--json-output`] name and path to assign to a JSON output results file
- **Optional:** `-p` [`--password`] password needed to open a protected PDF input file
- **Optional:** `-n` [`--nopath`] remove the system paths to the input PDF or JSON file

For each command line option, you can use either the short ("-i") or long ("--input") notation.

The only value in the command line statement that does not require a matching argument is the last one listed above, "--nopath" for removing the path from the input file name.

The command syntax for PDF Checker with the required values looks like this:

```
pdfchecker –-input test.pdf –-profile everything.json
```

In this command you are telling the system to inspect the PDF file called "test.pdf" using the JSON profile "everything.json."

PDF Checker will display the results of the review on your command line screen by default.

If you would like the product to export the results to an output file, use the "--output" option. Two kinds of output files are available, a text file and a JSON file. The text file is meant to be easy for you to read, while we designed the JSON output report to be machine-readable. You can either display the results report on your console or save it to an export file, or you can do both.

With the JSON output file, you can add the report results from PDF Checker to your own batch process. For example, you can collect and store the results for any PDF document you review with PDF Checker in your own database for later analysis.

These are the options that PDF Checker provides when you generate report output:

1. Display the text file report to the console (default, if you don't use the "- -output" option)
2. Save this content to an output text file
3. Display the JSON report content to the console
4. Save the JSON report content to an output JSON file
5. Save both kinds of output, human readable and machine readable, to a pair of output files, text and JSON
6. Display the JSON file output on the console and save the text content to an output text file
7. Display the text file output on the console and save the JSON output to an output JSON file

Note that you can't both display a report on your console and save the same content to an output file. It is possible to display both the text and JSON report content on the console, though a need for that seems unlikely. You could simply use PDF Checker to analyze the same PDF document twice instead.

To generate JSON output, you need to use the "-s" or "--json-output -" value to your command line statement.

You can also enter a password for opening a PDF document if the PDF input file is password protected.

Finally, if you don't want the path name for the input PDF document or of the JSON Profile file to appear in your output text file, or in the standard output provided by your command line tool, add the value "--nopath" to the end of your command line statement. You may want to suppress the place where you store your PDF input files or your JSON profile for security reasons.

The full command syntax for PDF Checker looks like this:

```
pdfchecker –-input test.pdf –-profile everything.json --output
PDFChecker_results.txt –-password mypassword --nopath
```

This command tells PDF Checker to export the results to a text file called PDFChecker_results.txt. Note that you need to add the prefix "--output" before the file name and the prefix "--password" before the password.

What if you wanted to export the output to a JSON file instead? The command syntax would look something like this:

```
pdfchecker –-input test.pdf –-profile everything.json –-json-
output PDFChecker_results.json –-password mypassword --nopath
```

Note the use of the "--json-output" option instead of "--output" in this command statement. If you want to display the JSON output on your command line monitor, you could use this command instead:

```
pdfchecker –-input test.pdf –-profile everything.json –-json-
output -
```

You can create both kinds of output files, text and JSON:

```
pdfchecker –-input test.pdf –-profile everything.json –-json-
output PDFChecker_results.json –-output PDFChecker_results.txt
```

You can display the JSON output on your console and save the same output as a text file:

```
pdfchecker –-input test.pdf –-profile everything.json –-json-
output - –-output PDFChecker_results.txt
```

Alternately, you could display the text output on your console and save the same output as a JSON formatted file:

```
pdfchecker –-input test.pdf –-profile everything.json --output -
–-json-output PDFChecker_results.json
```

The Windows installation program for PDF Checker adds the location of the PDF Checker executable to the PATH in the Windows Environment Variables. That means that you can run the executable, pdfchecker.exe, from anywhere.

You don't need to include a path name for any of the files mentioned in the command statement if the input file and profile are stored in the same directory as the executable pdfchecker.exe, and if you want to save the PDF Checker results file to the same directory. But you of course might want to draw an input file from one directory and save the output to another, and maybe store your JSON Profile files in a third.

In that event you need to provide the path as well as the file name:

```
pdfchecker –-input C:\Datalogics\CheckerFiles\AnnualReport2016.pdf –-
profile everything.json –-output C:\Datalogics\PDFChecker_results.txt
```

If any of the file or path names include spaces, use quotes around the name:

```
pdfchecker –-input C:\Datalogics\CheckerFiles\AnnualReport2016.pdf –-
profile everything.json –-output "C:\Datalogics\PDFChecker results.txt"
```

And you can provide a path name for the JSON profile file, just as you would for the input PDF document and the output results file:

```
pdfchecker –-input C:\Datalogics\CheckerFiles\AnnualReport2016.pdf –-
profile C:\Datalogics\JSONProfiles\everything.json –-output
C:\Datalogics\PDFChecker_results.txt
```

If you *don't* enter a path name for the JSON profile file as part of the "--profile" value, PDF Checker will look for the JSON profile file to use. It will first search the CheckerProfiles folder, which is part of the directory structure for your PDF Checker software installation package and where the standard JSON Profile file, everything.json, the file we provide with the software, is stored. If the file is not in the

CheckerProfiles directory, PDF Checker will treat the "--profile" input value in the command statement as a path to the profile and try to load the profile from that location.

This makes the product easier for you to use after you install it. All you need to do is provide the name of the JSON Profile file. PDF Checker can find the file for you.

You can use the JSON profile we provide, everything.json, from the command line, like this:

```
pdfchecker –-input test.pdf –-profile everything.json --output
PDFChecker _results.txt –-password mypassword --nopath
```

The JSON profile file

This is what the everything.json profile file looks like. The JSON content below includes every possible setting for PDF Checker.


```
{
    "general": {
        "unable-to-open": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Cannot be opened/not valid PDF",
            "abort-remaining-checks": "on"
        },
        "password-protected": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Requires password for opening",
            "abort-remaining-checks": "on"
        },
        "contains-owner-password": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains owner password",
            "abort-remaining-checks": "off"
        },
        "xfa-type": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "XFA document",
            "abort-remaining-checks": "off"
        },
        "acroforms-type": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "AcroForms document (no XFA)",
            "abort-remaining-checks": "off"
        },
        "pdf-v2": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "PDF 2.0 document",
            "abort-remaining-checks": "off"
        },
        "contains-signature": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Signed document",
            "abort-remaining-checks": "off"
        },
        "does-not-conform-to-claimed-pdfa-type": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Is not PDF/A conformant, see ‘claimed-pdfa-type’ in the
                               Summary section for more details.",
            "abort-remaining-checks": "off"
        },
```

```
        "claims-pdfx-conformance": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Claims PDF/X compliance",
            "abort-remaining-checks": "off"
        },
        "claims-pdfe-conformance": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Claims PDF/E compliance",
            "abort-remaining-checks": "off"
        },
        "claims-pdfvt-conformance": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Claims PDF/VT compliance",
            "abort-remaining-checks": "off"
        },
        "claims-pdfua-conformance": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Claims PDF/UA compliance",
            "abort-remaining-checks": "off"
        },
        "born-digital": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Document was born digital. It was produced from PDF
                               authoring software and so it may contain text, images,
                               tables, forms, and other objects. These types of PDFs
                               typically do not require OCR.",
            "abort-remaining-checks": "off"
        },
        "image-only": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Document was not born digital and only consists of images.
                               It may have been produced by scanning a document for
                               instance. These types of PDFs are good candidates for
                               OCR.",
            "abort-remaining-checks": "off"
        },
        "damaged": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Damaged document",
            "abort-remaining-checks": "off"
        },
        "tagged-pdf": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Document contains tagged content",
            "abort-remaining-checks": "off"
        }
    },
    "cleanup": {
```

```
        "suboptimal-compression": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains conservatively compressed streams",
            "abort-remaining-checks": "off"
        }
    },
    "fonts": {
        "found-non-extractible-text": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Uses fonts that do not allow text to be extracted",
            "abort-remaining-checks": "off"
        },

        "uses-fonts-not-embedded": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Uses fonts not embedded in document",
            "abort-remaining-checks": "off"
        },

        "uses-base14fonts-not-embedded": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Uses Base 14 fonts not embedded in document",
            "abort-remaining-checks": "off"
        },
        "uses-fonts-fully-embedded": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Uses fonts fully embedded in document",
            "abort-remaining-checks": "off"
        },
        "fontdescriptor-missing-fields": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "FontDescriptor has missing required fields",
            "abort-remaining-checks": "off"
        },
        "fontdescriptor-missing-capheight": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "FontDescriptor is missing potentially required
                               CapHeight field",
            "abort-remaining-checks": "off"
        }
    },
    "objects": {
        "contains-javascript-actions": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains JavaScript actions",
            "abort-remaining-checks": "off"
        },
        "contains-thumbnails": {
            "check": "on",
```

```
            "report-as-error": "off",
            "report-message": "Contains page thumbnail images",
            "abort-remaining-checks": "off"
        }
    },
    "userdata": {
        "contains-annots": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains annotations",
            "abort-remaining-checks": "off"
        },
        "contains-annots-not-for-viewing": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains annotations that are set as invisible
                               for viewing",
            "abort-remaining-checks": "off"
        },
        "contains-annots-not-for-printing": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains annotations that are set as invisible
                               for printing",
            "abort-remaining-checks": "off"
        },
        "contains-annots-without-normal-appearances": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Contains annotations without default normal appearances.
                               These may not be displayed correctly by all PDF viewers",
            "abort-remaining-checks": "off"
        },
        "contains-optional-content": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains optional content (layers)",
            "abort-remaining-checks": "off"
        },
        "contains-transparency": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains transparency",
            "abort-remaining-checks": "off"
        },
        "contains-private-data": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains application private data",
            "abort-remaining-checks": "off"
        },
        "contains-metadata": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains metadata",
            "abort-remaining-checks": "off"
        },
```

```
        "contains-embedded-files": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains embedded files",
            "abort-remaining-checks": "off"
        }
    },
    "images": {
        "color": {
            "resolution-too-low": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "Low resolution color image(s) present",
                "abort-remaining-checks": "off",
                "trigger-dpi": 150
            },
            "resolution-too-high": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "High resolution color image(s) present",
                "abort-remaining-checks": "off",
                "trigger-dpi": 600
            },
            "uses-jpeg2000-compression": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "Color image(s) using JPEG2000 compression",
                "abort-remaining-checks": "off"
            },
            "image-depth": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "16-bit/channel color image(s) present",
                "abort-remaining-checks": "off"
            }
        },
        "grayscale": {
            "resolution-too-low": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "Low resolution gray image(s) present",
                "abort-remaining-checks": "off",
                "trigger-dpi": 150
            },
            "resolution-too-high": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "High resolution gray image(s) present",
                "abort-remaining-checks": "off",
                "trigger-dpi": 600
            },


            "uses-jpeg2000-compression": {
                "check": "on",
                "report-as-error": "off",
                "report-message": "Grayscale image(s) using JPEG2000 compression",
```

```
            "abort-remaining-checks": "off"
        }
    },
    "monochrome": {
        "resolution-too-low": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Low resolution monochrome image(s) present",
            "abort-remaining-checks": "off",
            "trigger-dpi": 200
        },
        "resolution-too-high": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "High resolution monochrome image(s) present",
            "abort-remaining-checks": "off",
            "trigger-dpi": 1200
        },
        "uses-jbig2-compression": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Monochrome image(s) using JBIG2 compression",
            "abort-remaining-checks": "off"
        }
    },
    "alternate-images": {
        "check": "on",
        "report-as-error": "off",
        "report-message": "Alternate image(s) present",
        "abort-remaining-checks": "off"
        }
    }
}
```

Turning options on or off

You can create your own JSON profile by editing the JSON profile we provide and removing options you don't want. Or you can simply turn an option off. Set the "check" value to "off":

```
"objects": {
    "contains-javascript-actions": {
        "check": "off",
        "report-as-error": "off",
        "report-message": "Contains JavaScript actions",
        "abort-remaining-checks": "off"
```

Abort remaining checks

The "abort-remaining-checks" value appears for every setting. You can set up PDF Checker so that it stops its analysis of a given PDF document if it encounters a condition that you select. This saves processing time; PDF Checker abandons the current document if it finds a critical error. If you are running PDF Checker as a batch process, it will move on to scan the next PDF file in the directory.

Two conditions will cause the PDF Checker review process to stop processing immediately whether the "abort-remaining-checks" condition is turned on or not.

- **unable-to-open**. The document has an invalid format and cannot be opened, or it is not really a PDF file, or the file is named in the command line statement but is not in the directory.
- **password-protected**. The document requires a password to open and view, but the password was not provided with the command line statement.

A third condition might cause the review process to stop immediately if the PDF document cannot be repaired.

- **damaged**. The document is corrupted and PDF Checker cannot open the file.

  PDF Checker includes a document repair feature. If PDF Checker finds it cannot open a PDF document, it will run the repair process for the file, and store an updated version of the file in memory with those changes applied, but without changing the file itself. A similar feature in Adobe Acrobat allows a user to save a repaired document and apply those repairs. PDF Checker does not make any changes to the initial document or allow the saving of a repaired document. Other Datalogics products can be used for that.

  If PDF Checker still can't open the PDF document after repairing it, the system will stop processing the document and mark it as damaged. If PDF Checker can open the repaired file, it will run through its normal review process, but the review might not complete successfully. If one of the reviews fails, PDF Checker will stop processing the document, and it will only report on the reviews that completed. For example, if PDF Checker can identify and list the fonts in a repaired PDF document but fails when seeking to process the images in that document, checks made during processing images will be reported as aborted.

Suppose you set the “abort-remaining-checks” value to “on” for the “contains-annots-without-normal-appearances” condition, like this:

```
        "contains-annots-without-normal-appearances": {
            "check": "on",
            "report-as-error": "off",
            "report-message": "Contains annotations without default normal appearances.
                               These may not be displayed correctly by all PDF viewers",
            "abort-remaining-checks": "on"
```

In this example, the first time PDF Checker finds annotations without normal appearances in a PDF document, it stops all further processing for that document. Since PDF Checker can run multi-threaded, however, and thus complete multiple checks at one time, the actual remaining checks that are not processed can vary, depending on when the abort signal is received. For example, one thread could be checking fonts in a PDF document while another is looking at images. The thread checking fonts might not be finished, or where it breaks off might vary from one PDF document to another, when PDF Checker abandons the PDF file. As a result, the output results might vary, too. The check that causes the process to abort is shown in the output report.

Reporting an issue as an error or information

PDF Checker evaluates a PDF document for each setting included in you JSON profile and generates a record for each result. If you were to export your results to a text file, part of the report might look like this:

```
Fonts Results
    Errors:
        Uses fonts not embedded in document:
            Arial (1 instance)
            Arial,Bold (1 instance)
            CourierStd (1 instance)
            Meiryo (1 instance)
            Murdana (3 instances)
            TimesNewRoman (1 instance)
            TimesNewRoman,Italic (1 instance)
```

Note the options in the JSON Profile to report as error, and the error message:

```
    "fonts": {
        "uses-fonts-not-embedded": {
            "check": "on",
            "report-as-error": "on",
            "report-message": "Uses fonts not embedded in document",
            "abort-remaining-checks": "off"
```

You can use these settings to flag a result specifically as an error in the output results. Set “report-as-error” to “on” and the result will appear under the heading “Errors” in the export file, as shown above. You can also edit the message that appears by changing the value for “report-message.” You could change the error message from this:

```
            "report-as-error": "on",
            "report-message": "Uses fonts not embedded in document",
```

To this:

```
            "report-as-error": "on",
            "report-message": "These fonts are not found in this PDF document",
```

Description of JSON profile parameters

general: unable-to-open

This is not a valid PDF document, or it has been corrupted to the point that it cannot be displayed in a browser or viewing tool.

general: password-protected

The file cannot be opened without a password. To open the file, provide the password in a command line argument.

general: contains-owner-password

When you create a PDF document you can restrict the ability of others to work with that document and add a PDF Owner Password to the file to secure those settings. Other users will be able to open and read your PDF document, but without this password they can't change your restrictions. With the Owner Password you can allow others to open and read your document but stop them from printing the file, copying content, adding changes or comments, adding or extracting pages or graphics, signing the document, or making other changes.

general: xfa-type

XFA, or XML Forms Architecture, is a set of proprietary XML specifications for use with web forms. XFA forms are saved internally in PDF files, and the standard is owned by Adobe Inc.

general: acroforms-type

Check for presence of AcroForm content in a PDF Document. Most PDF forms documents use Acroforms rather than XFA, because Acroforms are compatible with a much wider range of software applications, as well as with Acrobat itself.

general: pdf-v2

The PDF format was published as an open file format by the International Organization for Standardization (ISO) in 2008. Version 2.0 of PDF was released in July of 2017. PDF Checker can identify if a PDF document was created recently, as a PDF 2.0 document.

general: contains-signature

A PDF document can contain one or more digital signatures, and these signatures can be verified by a vendor known as a Certifying Authority. If a PDF document has a certified digital signature it can be used as a legal document. It is also possible to lock a signed PDF document against any further changes.

general: does-not-conform-to-claimed-pdfa-type

A PDF/A, or PDF Archive document, is a type of PDF file that is designed to be stored so that it can be accessed for many years to come. PDF Checker will determine if a PDF document *claims* PDF/A compliance and if that claim is valid. It is possible for a PDF document to be identified as compliant when in fact it's not. PDF Checker will also provide the type of PDF/A that is claimed in the PDF document:

PDF/A-1b	Basic conformance for visual appearance.
PDF/A-1a	Matches PDF/A-1a, but also provides for accessibility for people with disabilities. This allows a person to work with a PDF document while using assistive software, such as viewing tools that can read a document out loud.
PDF/A-2b	Basic conformance with archival standards but revised for later versions of the PDF format. PDF/A-2 includes support for OpenType fonts, layers, attachments (which must also be PDF/A compliant) and JPEG 2000 image compression.
PDF/A-2u	Matches PDF/A-2b but also requires that all text in the document have Unicode mappings.
PDF/A-3b	Matches PDF/A-2b, except that it is possible to embed any kind of file in the PDF document. For example, with PDF/A-3 a user can save a XML, CSV, CAD, spreadsheet, or other type of file in the PDF document and be compliant. The file embedded in the PDF/A-3 does not need to PDF/A compliant.
PDF/A-3u	Matches PDF/A-3b, but also requires that all text in the document have Unicode mapping.

Note: PDF Checker still supports the "claims-pdfa-conformance" check for backwards compatibility. We recommend that you use "does-not-conform-to-claimed-pdfa-type" instead, as this check determines the type of PDF/A document claimed and can verify if the document does in fact conform to that standard.

general: claims-pdfx-conformance

PDF/X refers to PDF Graphics document, commonly used for graphics exchanges and printing art. PDF/X graphics files can define ICC color profiles, support spot colors, and access external graphics content.

PDF Checker can determine if a PDF document claims to be PDF/X compliant, but not if the PDF/X conformance is actually valid.

general: claims-pdfe-conformance

PDF/E refers to PDF Engineering documents, used for engineering and construction workflows.

PDF Checker can determine if a PDF document claims to be PDF/E compliant, but not if the PDF/E conformance is actually valid.

general: claims-pdfvt-conformance

A PDF/VT document uses variable and transactional printing, or Variable Data Printing (VDP). In VDP, elements such as text, graphics and images may be changed from one printed piece to the next using information drawn from an external file or database, and without stopping the printing process.

PDF Checker can determine if a PDF document claims to be PDF/VT compliant, but not if the PDF/VT conformance is actually valid.

general: claims-pdfua-conformance

PDF/UA refers to Universal Accessibility. PDF/UA documents are designed to be used by people with disabilities, in particular, to make them easier to use by people who are visually impaired. PDF Checker can determine if a PDF document claims to be PDF/UA compliant, but not if the PDF/UA conformance is actually valid.

general: damaged

PDF Checker found that the PDF document was corrupted, and on its first attempt the system was not able to open it. In response PDF Checker sought to repair the file so that it could be opened.

After repairing the file PDF Checker seeks to open the file again. If PDF Checker still can't open the PDF document, no further processing is possible. The report will simply indicate that the document is damaged.

If PDF Checker can repair and open a damaged document, the system will process as many review checks as possible and report on any review steps that complete properly. If PDF Checker cannot complete any checks, the system will stop processing the document and report those review steps as aborted. The failed steps might be due to the document being damaged.

general: born-digital
general: image-only

Most PDF documents are generated using a PDF authoring tool, such as saving a Word document or an Excel spreadsheet as a PDF output file or creating a PDF document using Adobe Acrobat. This standard variety of PDF document features the content that PDF documents are known for, including text, graphics, annotations, embedded images, hyperlinks, bookmarks, and other objects.

But it is also possible to create a PDF document by feeding a print document through a scanner and saving the output as PDF. In this case the PDF document will only feature a series of embedded graphical image files, such as PNG, TIF, or JPG files, one for each page. The PDF won't have any text or other features common to PDF documents, except for some metadata.

A user won't see a significant difference between these two kinds of PDF documents when opening them in Adobe Reader or some other viewing tool. But you can use PDF Checker to review a PDF document to determine if it is a standard PDF or was created using a scanner. This would be particularly useful to know if you want to process a PDF document using Optical Character Recognition (OCR) technology. If PDF Checker finds that a PDF document is a standard PDF with 10 pages of text and no images or graphics, there would be nothing to gain by running the file through an OCR process. But if PDF Checker finds that a PDF was created by scanning a print document, and features ten pages of embedded graphics files, processing that file with OCR could identify the text in these images and then render that text content to an HTML or XML export file.

general: tagged-pdf

Checks for presence of structure tags in the input document. A tagged PDF document contains information to describe items such as headers and other content on a page. Tagging is generally used with a PDF document to meet accessibility requirements. For example, tags in a PDF document might be placed so that text, headings, footnotes, and other content in the document can be interpreted by a screen reading software tool.

cleanup: suboptimal-compression

A data stream in a PDF document contains text, an image, or an object, with instructions on how the content will be rendered on the page. These data streams can be compressed in a document to make the PDF smaller and more portable. This check looks for data streams in the input document that are not compressed, or that are using a simple algorithm that is not as efficient in compression, such as ASCII, or Run Length, or LZW.

PDF Optimizer can be used to improve the compression used in a PDF document to reduce the size of the file.

fonts: found-non-extractible-text

Checks for the presence of fonts used in the input PDF that do not allow for text to be extracted in a standard way that a PDF processor would be expected to understand. This can cause problems in working with the PDF document. For example, if text cannot be extracted from a PDF, a PDF viewer will not be able to allow a user to search for that text within the document.

fonts: uses-fonts-not-embedded
fonts: uses-base14fonts-not-embedded
fonts: uses-fonts-fully-embedded

It is a best practice when working with PDF documents to embed, or save, every font used in a PDF document in the document itself. That way, a viewing tool (like Acrobat) does not have to look for a font stored on the local system or choose a substitute font. Use this setting to find out if a PDF is using fonts that are not embedded. Removing an embedded font file from a PDF document can make the document smaller, but this practice can also make the PDF load more slowly. It might also change the appearance of the file if the viewing software cannot find a font it needs on the local machine nor a suitable substitute.

PDF Checker looks in the /FontDescriptor directory, or the font dictionary, within the PDF document, to identify the fonts that are in use in that document. Then, it looks to see if those same font files are embedded in that document, or if the viewer will need to access a font from the host machine.

Fonts that are not embedded in the PDF document, either Base 14 fonts or otherwise, and fonts that are embedded, can be listed in the results.

PDF Optimizer can be used to subset fonts that are fully embedded in PDF files. Subsetting fonts can reduce the size of PDF files by reducing the fonts that are in those PDFs to only those characters that are actually used in the PDF. However, subsetting can impact the ability to later edit the text within a PDF file. Subsetting should be used when a PDF is intended for final-form distribution rather than for editing.

fonts: fontdescriptor-missing-fields
fonts: fontdescriptor-missing-capheight

A font descriptor describes the characteristics of a font, as opposed to the widths and characteristics of individual glyphs (characters) within that font set. Font descriptor values include the name of the font, the angle in degrees used for creating italic characters, the maximum height above the baseline that a glyph can reach in this font (ascent) and the maximum depth below that it can reach (descent), and a variety of other values.

If one of the required font descriptor settings is not included in the font descriptor dictionary for a PDF document, PDF Checker can determine that and list the missing values. If a font descriptor value is not provided the PDF document may not work properly in some applications.

The CapHeight is the coordinate showing the placement of the tops of flat capital letters, such as T or R, as measured from the baseline. It is required for all fonts that have Latin characters except for Type 3 fonts (that use PostScript). But as it is hard to determine if a given font has Latin characters, it is possible with a standard font descriptor search for the CapHeight value to be overlooked. To avoid that, PDF Checker searches for CapHeight separately.

objects: contains-javascript-actions

This PDF document contains blocks of JavaScript code that actions that may alter appearance of the document. Common JavaScript actions in PDF documents include submitting a form (a Submit button), accessing a web site from a web address, or sending the document to a printer.

PDF Optimizer can be used to remove JavaScript actions from PDF documents. This can reduce the size of the PDF file and increase compatibility across PDF viewers and processors.

objects: contains-thumbnails

Thumbnail images are used to preview pages in a PDF document and appear in a panel on the left side of the viewer window. A user could scroll through a series of thumbnails to find a page he or she is looking for.

PDF Optimizer can be used to remove thumbnails from PDF documents. This can reduce the size of the PDF file.

userdata: contains-annots

Annotations are changes you can add to a PDF document, such as notes, highlighted text, file attachments, crossed out text, and text callout boxes.

userdata: contains-annots-not-for-viewing

Annotations can be added to a PDF document and hidden, so that they do not appear when the document is opened in a viewing tool.

userdata: contains-annots-not-for-printing

Annotations can be added to a PDF document so that they appear on the page in a viewing tool but are not included when the document is sent to a printer.

userdata: contains-annots-without-normal-appearances

Every annotation included in a PDF document features an optional entry that describes what the annotation will look like when the document is rendered in a viewer. Generally, this value is not provided, so if the PDF is opened in Adobe Acrobat, Acrobat will fill in the appearance, based on what the value should be. When you open a PDF document in Adobe Acrobat or Adobe Reader and this viewer fills in the appearance value, you will be prompted to save the file when you close it. If you do save the file, the annotation appearance is made a part of the updated PDF document. This is the normal annotation appearance.

userdata: contains-optional-content

Optional content in a PDF document are layers of graphics objects or annotations that can be made to appear or disappear when the PDF is opened in Adobe Acrobat or Reader, using the Layers pane.

PDF Optimizer can be used to remove these extra layers from PDF documents. This can reduce the size of PDF files and increase compatibility across PDF viewers and processors.

userdata: contains-transparency

It is possible to stack objects, such as graphics, images, text boxes, and form fields, on top of each other on a PDF document. These objects can be partially or fully transparent, and thus can interact in various ways with objects behind them. If a set of transparencies are stacked in a PDF file, each one contributes to the final result that appears on the page, such as the colors blending together into a final color that appears.

PDF Optimizer can be used to flatten the transparency in a PDF document. Flattening a transparency substitutes a representation of transparent content in that document, merging and blending together the appearance of transparent content. This can be used to improve the compatibility of PDFs across PDF viewers and processors, and to decrease the amount of time required to render and print PDF pages using transparency.

userdata: contains-private-data

Some applications, like Adobe Illustrator, add their own unique values to a PDF document when generating that document. These values are useful to the original software product if the PDF is opened and edited in that product again.

PDF Optimizer may be used to remove private data from PDF files. Removing private data can reduce the size of a PDF document, though it will remove the ability for some applications to edit these PDF files. This should be used for PDF documents that are intended for final-form distribution, rather than for PDFs that are intended for editing.

userdata: contains-metadata

Most PDF documents store information that describes that document, such as the author, creation date, and the software used to generate the file.

PDF Optimizer may be used to remove much of the document information and metadata from PDF files, to reduce the size of a PDF document. A minimal set of information will be maintained to ensure maximum compatibility with PDF viewers and processors.

userdata: contains-embedded-files

PDF documents can hold other files that are embedded or attached in that document, including other PDF documents, email messages, spreadsheets, graphics files, and the like.

PDF Optimizer may be used to remove embedded files and attachments from PDF documents. This can reduce the size of PDF files significantly, for PDF files where these attachments are not desired or are not intended to be used.

images: color

PDF Optimizer provides a variety of different methods for optimizing images that can reduce the size of color images. These optimizations include the ability to downsample (reduce the resolution of) images, the ability to change the compression of color images, and the ability to reduce the color depth of images. These optimizations can be controlled specifically for color images.

Also, PDF Optimizer can be used to perform color conversion on color images in PDF files. This can be used to normalize the color representations used in a PDF document for faster processing, improved compatibility, and decreased file sizes. And color conversion may also be used to transform PDFs into grayscale renditions. This can dramatically speed up printing of many PDF files when sent to black and white printers.

resolution-too-low

Number of low-resolution color images present in the document. PDF Checker determines the resolution of each image in the PDF document, and any color image with a resolution below this Trigger DPI value is counted as a low-resolution image. This Trigger value parameter defaults to 150 DPI for color images.

resolution-too-high

Number of high-resolution color images present in the document. Any color image PDF Checker finds with a resolution greater than this Trigger DPI value is counted as a high-resolution image. The Trigger value parameter defaults to 600 DPI for color images.

uses-jpeg2000-compression

Number of color images in the document using JPEG2000 compression. JPEG compression is a compression format used for rendering photographs as image files. It is also known as DCT, Discrete Cosine Transform.

image-depth

PDF Checker lists the number of 16-bit color images found in the PDF document. Image depth refers to the number of bits needed to store color for each pixel in a graphic. Color graphics are often 8-bit, but higher quality images are 16-bit or more. A color graphic with 16-bit image depth will usually render better on a screen or when printing, but the image is also a lot larger, making the PDF document a lot larger as well.

images: grayscale

PDF Optimizer provides a variety of different methods for optimizing images that can reduce the size of grayscale images. These optimizations include the ability to downsample (reduce the resolution of) images and the ability to change the compression of grayscale images. These optimizations can be controlled specifically for grayscale images.

resolution-too-low

Number of low-resolution grayscale images present in the document. PDF Checker determines the resolution of each image in the PDF document, and any grayscale image with a resolution below this Trigger DPI value is counted as a low-resolution image. This Trigger value parameter defaults to 150 DPI for grayscale images.

resolution-too-high

Number of high-resolution grayscale images present in the document. Any grayscale image PDF Checker finds with a resolution greater than this Trigger DPI value is counted as a high-resolution image. This Trigger value parameter defaults to 600 DPI for grayscale images.

uses-jpeg2000-compression

Number of grayscale images in the document using JPEG2000 compression. JPEG compression is a compression format used for rendering photographs as image files. It is also known as DCT, Discrete Cosine Transform.

images: monochrome

PDF Optimizer provides a variety of different methods for optimizing images that can reduce the size of monochrome images. These optimizations include the ability to downsample (reduce the resolution of) images and the ability to change the compression of monochrome images. These optimizations can be controlled specifically for monochrome images.

resolution-too-low

Number of low-resolution monochrome images present in the document. PDF Checker determines the resolution of each image in the PDF document, and any monochrome image with a resolution below this Trigger DPI value is counted as a low-resolution image. This Trigger value parameter defaults to 200 DPI for monochrome images.

resolution-too-high

Number of high-resolution monochrome images present in the document. Any monochrome image PDF Checker finds with a resolution greater than this Trigger DPI value is counted as a high-resolution image. This Trigger value parameter defaults to 1200 DPI for monochrome images.

uses-jpeg2000-compression

Number of monochrome images in the document using JBIG2 compression. JBIG2 is a compression algorithm designed for binary images, or images where each pixel can only have one of two possible colors. For PDF Checker JBIG2 is used for black and white images.

images: alternate-images

A PDF document can be set up to specify alternate images, or multiple versions of one image within the same document. These images can be used to meet different needs. For example, a PDF could present one image with a lower resolution for display on a monitor, and an alternate image with a higher resolution to use when the PDF document is sent to a printer. These images can make a PDF document very large. Today alternate images are rarely used.

PDF Optimizer may be used to remove alternate images from PDF files. Removing alternate images can dramatically reduce the size of PDF files that contain these.

Sample Results Text File Output: Successful Review

This is what a complete results report would look like when each of the fields controlling the output is specified and PDF Checker completes the review successfully.

Note the section at the top of the results output, the CHECKER_SUMMARY.

This is a summary list of the items that PDF Checker found when reviewing a PDF document, intended to be machine readable. But you might prefer to use the JSON output format instead.

You could set up a batch process to review a set of PDF documents stored in a server directory automatically, one by one. Then, you could add a step to your batch code that would identify the PDF documents that PDF Checker found with issues or problems, create an input JSON file listing those issues or problems, and send that JSON file list to PDF Optimizer. PDF Optimizer could then optimize each PDF document based on the items found by PDF Checker. Note that for documents that PDF Optimizer can improve, a statement appears, "canBeOptimized," to make it easy detect documents that are candidates for optimization. The software also provides the size of the PDF source document, and a machine readable value, "sizeInBytes." This allows you to scan the results to make decisions on document processing based on file size. For example, a PDF document that is only 10 KB probably isn't worth optimizing.

Note that the metadata for the PDF document is provided at the top of the report. The "Trapped" parameter refers to the prepress workflow when printing in color. Trapping a document governs how the ink will appear on the page; colors are trapped by adjusting the shape of objects as they are printed on the page to avoid gaps from appearing. Trapping can be enabled by some software products that generate PDF documents, such as Adobe InDesign.

PDF Checker will determine if the PDF document claims PDF/A compliance, and provide the type of PDF/A file. If in fact the file is not PDF/A compliant, an error message will appear in the output report.

```
PDF Checker 2.2.0  Copyright 2018-2022 Datalogics, Inc. All Rights Reserved

Mon Dec 5 16:49:23 2022

JSON Profile: everything.json

Input Document: DocumentationExample.pdf

Number of Pages: 1894

Language: None

PDF Version 1.7

Claimed PDF/A Type: PDF/A-2a
Conforms to Claimed PDF/A Type: False

Author: John Smith
Creation Date: 2020-Sep-08 13:13:58
Creator: Acrobat PDFMaker 15 for Word
Keywords:
Modification Date: 2020-Sep-08 13:14:04
Producer: Adobe PDF Library 18.0
Subject:
Title:
Trapped:

File Size: 4.2 MB
```

```
<<=CHECKER_SUMMARY_START=>>
general:born-digital
general:does-not-conform-to-claimed-pdfa-type
general:tagged-pdf
userdata:contains-annots
userdata:contains-annots-not-for-printing
userdata:contains-metadata
userdata:contains-transparency
fonts:found-non-extractible-text
fonts:uses-base14fonts-not-embedded
fonts:uses-fonts-not-embedded
cleanup:suboptimal-compression
sizeInBytes:402518
canBeOptimized
<<=CHECKER_SUMMARY_END=>>

Optimization Assessment
    Document can be optimized with PDF Optimizer – see details below

General Results
    Errors:
        Is not PDF/A conformant, see 'claimed-pdfa-type' in the Summary section for more
details.:
            Total: (1 instance)
    Information:
        Document was born digital.  It was produced from PDF authoring software and so it may
contain text, images, tables, forms, and other objects.  These types of PDFs typically do not
require OCR.
        Is not PDF/A conformant:
            Total: (1 instance)
        Document contains tagged content
    Checks Completed:
        acroforms-type
        born-digital
        claims-pdfe-conformance
        claims-pdfua-conformance
        claims-pdfvt-conformance
        claims-pdfx-conformance
        contains-owner-password
        contains-signature
        damaged
        does-not-conform-to-claimed-pdfa-type
        image-only
        password-protected
        pdf-v2
        tagged-pdf
        unable-to-open
        xfa-type
    How To Optimize:
        Document claims to be PDF/A-2a conformant, but it is not, try using PDF Optimizer to
convert to a compliant PDF/A document. (1 instance)

Userdata Results
    Errors:
        None
    Information:
        Contains annotations:
            SubType: Link (21 instances)
        Contains annotations that are set as invisible for printing:
            SubType: Link (21 instances)
        Contains metadata:
            SubType: XML, Update region size: 2048 (1 instance)
            Total: (1 instance)
        Contains transparency:
            Total: (1 instance)
    Checks Completed:
        contains-annots
        contains-annots-not-for-printing
        contains-annots-not-for-viewing
        contains-annots-without-normal-appearances
        contains-embedded-files
```

```
        contains-metadata
        contains-optional-content
        contains-private-data
        contains-transparency
    How To Optimize:
        Annotations can be removed using PDF Optimizer to save space. (21 instances)
        Annotations not intended for printing can be removed using PDF Optimizer to save space.
(21 instances)
        Metadata can be removed using PDF Optimizer to save space. (1 instance)
        XMP Metadata padding can be removed using PDF Optimizer to save space. (1 instance)
        Transparency (which is not universally supported by PDF Viewers) can be flattened using
PDF Optimizer. (1 instance)

Fonts Results
    Errors:
        Uses Base 14 fonts not embedded in document:
            Arial-BoldMT (1 instance)
            ArialMT (1 instance)
            CourierNewPSMT (1 instance)
        Uses fonts not embedded in document:
            CenturyGothic (1 instance)
            CenturyGothic-Bold (1 instance)
    Information:
        Uses fonts that do not allow text to be extracted:
            HSODIF+SymbolMT, SubType: Type0 (1 instance)
    Checks Completed:
        fontdescriptor-missing-capheight
        fontdescriptor-missing-fields
        found-non-extractible-text
        uses-base14fonts-not-embedded
        uses-fonts-fully-embedded
        uses-fonts-not-embedded

Objects Results
    Errors:
        None
    Information:
        None
    Checks Completed:
        contains-javascript-actions
        contains-thumbnails

Cleanup Results
    Errors:
        None
    Information:
        Contains conservatively compressed streams:
            Uncompressed: (1 instance)
    Checks Completed:
        suboptimal-compression

Image Results
    Errors:
        None
    Information:
        None
    Checks Completed:
        alternate-images

    Color Images
    Errors:
        None
    Information:
        None
    Checks Completed:
        image-depth
        resolution-too-high
        resolution-too-low
        uses-jpeg2000-compression

    Grayscale Images
```

```
Errors:
    None
Information:
    None
Checks Completed:
    resolution-too-high
    resolution-too-low
    uses-jpeg2000-compression

Monochrome Images
Errors:
    None

Information:
    None
Checks Completed:
    resolution-too-high
    resolution-too-low
    uses-jbig2-compression
```

Sample Results Text File Output: Abort Results Output

This is what the results report would look like if one of the fields has the “abort-remaining-checks” value turned on. In this example, the value is "contains-annots-without-normal-appearances."

The first instance of the check will stop all further processing. Since PDF Checker is multi-threaded and doing multiple checks at once, the actual remaining checks that are not processed can vary when the signal is received. The check that triggered the abort is shown in the report below.

```
PDF Checker 2.2.0  Copyright 2018-2022 Datalogics, Inc. All Rights Reserved

Mon Dec 5 16:59:21 2022

JSON Profile: everything.json

Input Document: DocumentationExampleWithAbort.pdf

Number of Pages: 1894

Language: None

File Size: 4.2 MB

<<=CHECKER_SUMMARY_START=>>
userdata:contains-annots-without-normal-appearances
userdata:contains-metadata
sizeInBytes: 4211884
<<=CHECKER_SUMMARY_END=>>

General Results
    Errors:
        None
    Information:
        None
    Checks Completed:
        claims-pdfa-conformance
        contains-owner-password
        contains-signature
        damaged
        password-protected
        pdf-v2
        unable-to-open
        xfa-type

Userdata Results
    Errors:
        None
    Information:
        Contains metadata:
            SubType: XML, Update region size: 2048 (1 instance)
    Checks Completed:
        contains-metadata
        contains-optional-content
    Checks Aborted:
        contains-annots
        contains-annots-not-for-printing
        contains-annots-not-for-viewing
        contains-annots-without-normal-appearances (triggered abort)
        contains-embedded-files
        contains-private-data
        contains-transparency

Fonts Results
    Errors:
        None
    Information:
        None
    Checks Completed:
```

```
        None
    Checks Aborted:
        fontdescriptor-missing-capheight
        fontdescriptor-missing-fields
        uses-base14fonts-not-embedded
        uses-fonts-fully-embedded
        uses-fonts-not-embedded

Objects Results
    Errors:
        None
    Information:
        None
    Checks Completed:
        contains-javascript-actions
    Checks Aborted:
        contains-thumbnails

Cleanup Results
    Errors:
        None
    Information:
        None
    Checks Completed:
        None
    Checks Aborted:
        suboptimal-compression

Image Results
    Errors:
        None
    Information:
        None
    Checks Completed:
        None
    Checks Aborted:
        alternate-images

    Color Images
    Errors:
        None
    Information:
        None
    Checks Completed:
        None
    Checks Aborted:
        image-depth
        resolution-too-high
        resolution-too-low
        uses-jpeg2000-compression

    Grayscale Images
    Errors:
        None
    Information:
        None
    Checks Completed:
        None
    Checks Aborted:
        resolution-too-high
        resolution-too-low
        uses-jpeg2000-compression

    Monochrome Images
    Errors:
        None
    Information:
        None
    Checks Completed:
        None
    Checks Aborted:
```

```
resolution-too-high
resolution-too-low
uses-jbig2-compression
```

Sample Results JSON File Output: Successful Review

This is what a complete JSON results report file would look like when each of the fields controlling the output is specified and PDF Checker completes the review successfully.

You could set up a batch process to review a set of PDF documents stored in a server directory automatically, one by one. Then, you could add a step to your batch code that would identify the PDF documents that PDF Checker found with issues or problems, and send this JSON report output to PDF Optimizer, and PDF Optimizer could then optimize each PDF document based on the items found by PDF Checker.

Note that PDF Checker provides the size of the PDF source document, and a machine readable value, "sizeInBytes." This allows you to scan the results to make decisions on document processing based on file size. For example, a PDF document that is only 10 KB probably isn't worth optimizing.

Note that the metadata for the PDF document is provided at the top of the report. The "Trapped" parameter refers to the prepress workflow when printing in color. Trapping a document governs how the ink will appear on the page; colors are trapped by adjusting the shape of objects as they are printed on the page to avoid gaps from appearing. Trapping can be enabled by some software products that generate PDF documents, such as Adobe InDesign.

PDF Checker will determine if the PDF document claims PDF/A compliance, and provide the type of PDF/A file. If in fact the file is not PDF/A compliant, an error message will appear in the output report.


```
{
    "analysis-summary": {
        "can-be-optimized": true,
        "claimed-pdfa-type": PDF/A-1b,
        "conforms-to-claimed-pdfa-type": true,
        "date": "Mon Dec 5 11:07:33 2022",
        "errors": [
            "uses-base14fonts-not-embedded",
            "uses-fonts-not-embedded"
        ],
        "information": [
            "born-digital",
            "contains-annots",
            "contains-annots-not-for-printing",
            "contains-metadata",
            "contains-transparency",
            "found-non-extractible-text",
            "suboptimal-compression",
            "tagged-pdf"
        ],
        "input"C:\\path\\to\\input.pdf",
        "language": "None",
```

```
    "metadata": {
        "Author": "John Smith",
        "CreationDate": "2020-09-08T13:13:58",
        "Creator": "Acrobat PDFMaker 15 for Word",
        "Keywords": "",
        "ModDate": "2020-09-08T13:14:04",
        "Producer": "Adobe PDF Library 18.0",
        "Subject": "",
        "Title": "",
        "Trapped": ""
    },
    "number-of-pages": 1894,
    "pdf-checker-version": "2.2.0",
    "pdf-extension-level": 5,
    "pdf-major-version": 1,
    "pdf-minor-version": 7,
    "profile": "C:\\path\\to\\everything.json",
    "size-in-bytes": 402518,
    "triggered-abort": ""
},
"cleanup-results": {
    "checks-aborted": [],
    "checks-completed": [
        "suboptimal-compression"
    ],
    "errors": {},
    "how-to-optimize": {},
    "information": {
        "suboptimal-compression": [
            "Uncompressed: (1 instance)"
        ]
    }
},
"color-images": {
    "checks-aborted": [],
    "checks-completed": [
        "image-depth",
        "resolution-too-high",
        "resolution-too-low",
        "uses-jpeg2000-compression"
    ],
    "errors": {},
    "how-to-optimize": {},
```

```
        "information": {}
    },
    "fonts-results": {
        "checks-aborted": [],

        "checks-completed": [
            "fontdescriptor-missing-capheight",
            "fontdescriptor-missing-fields",
            "found-non-extractible-text",
            "uses-base14fonts-not-embedded",
            "uses-fonts-fully-embedded",
            "uses-fonts-not-embedded"
        ],
        "errors": {
            "uses-base14fonts-not-embedded": [
                "Arial-BoldMT (1 instance)",
                "ArialMT (1 instance)",
                "CourierNewPSMT (1 instance)"
            ],
            "uses-fonts-not-embedded": [
                "CenturyGothic (1 instance)",
                "CenturyGothic-Bold (1 instance)"
            ]
        },
        "how-to-optimize": {},
        "information": {
            "found-non-extractible-text": [
                "HSODIF+SymbolMT, SubType: Type0 (1 instance)"
            ]
        }
    },
    "general-results": {
        "checks-aborted": [],
        "checks-completed": [
            "acroforms-type",
            "born-digital",
            "claims-pdfe-conformance",
            "claims-pdfua-conformance",
            "claims-pdfvt-conformance",
            "claims-pdfx-conformance",
            "contains-owner-password",
            "contains-signature",
            "damaged",
```

```
            “does-not-conform-to-claimed-pdfa-type”,
            "image-only",
            "password-protected",
            "pdf-v2",
            "tagged-pdf",
            "unable-to-open",
            "xfa-type"
        ],
        "errors": {},
        "how-to-optimize": {},
        "information": {
            "born-digital": [
                "Document was born digital.  It was produced from PDF authoring software and so it may contain text, images, tables, forms, and other objects.  These types of PDFs typically do not require OCR."
            ],
            "tagged-pdf": [
                "Document contains tagged content"
            ]
        }
    },
    "grayscale-images": {
        "checks-aborted": [],
        "checks-completed": [
            "resolution-too-high",
            "resolution-too-low",
            "uses-jpeg2000-compression"
        ],
        "errors": {},
        "how-to-optimize": {},
        "information": {}
    },
    "image-results": {
        "checks-aborted": [],
        "checks-completed": [
            "alternate-images"
        ],
        "errors": {},
        "how-to-optimize": {},
        "information": {}
    },
    "monochrome-images": {
        "checks-aborted": [],
```

```
        "checks-completed": [
            "resolution-too-high",
            "resolution-too-low",
            "uses-jbig2-compression"
        ],
        "errors": {},
        "how-to-optimize": {},
        "information": {}
    },
    "objects-results": {
        "checks-aborted": [],
        "checks-completed": [
            "contains-javascript-actions",
            "contains-thumbnails"
        ],
        "errors": {},
        "how-to-optimize": {},
        "information": {}
    },
    "userdata-results": {
        "checks-aborted": [],
        "checks-completed": [
            "contains-annots",
            "contains-annots-not-for-printing",
            "contains-annots-not-for-viewing",
            "contains-annots-without-normal-appearances",
            "contains-embedded-files",
            "contains-metadata",
            "contains-optional-content",
            "contains-private-data",
            "contains-transparency"
        ],
        "errors": {},
        "how-to-optimize": {
            "contains-annots": [
                "Annotations can be removed using PDF Optimizer to save space. (21 instances)"
            ],
            "contains-annots-not-for-printing": [
                "Annotations not intended for printing can be removed using PDF Optimizer to save
space. (21 instances)"
            ],
            "contains-metadata": [
                "Metadata can be removed using PDF Optimizer to save space. (1 instance)",
```

```
                "XMP Metadata padding can be removed using PDF Optimizer to save space. (1
instance)"
            ],
            "contains-transparency": [
                "Transparency (which is not universally supported by PDF Viewers) can be
flattened using PDF Optimizer. (1 instance)"
            ]
        },
        "information": {
            "contains-annots": [
                "SubType: Link (21 instances)"
            ],
            "contains-annots-not-for-printing": [
                "SubType: Link (21 instances)"
            ],
            "contains-metadata": [
                "SubType: XML, Update region size: 2048 (1 instance)",
                "Total: (1 instance)"
            ],
            "contains-transparency": [
                "Total: (1 instance)"
            ]
        }
    }
}
```

Sample Results JSON File Output: Abort Results Output

This is what the JSON results report would look like if PDF Checker determines that the PDF source file is not in fact a PDF document at all, or if PDF Checker can't open the file.

The first instance of the check will stop all further processing. Since PDF Checker is multi-threaded and doing multiple checks at once, the actual remaining checks that are not processed can vary when the signal is received. The check that triggered the abort is shown in the report below.


```
{
    "analysis-summary": {
        "can-be-optimized": false,
        "date": "Mon Dec 5 12:08:07 2022",
        "errors": ["unable-to-open"],
        "information": [ ],
        "input": "C:\\path\\to\\input.pdf",
        "language": "None",
        "number-of-pages": 0,
        "pdf-checker-version": "2.2.0",
        "profile": "C:\\path\\to\\everything.json",
        "size-in-bytes": 3538813,
        "triggered-abort": "unable-to-open"
    },
    "cleanup-results": {
        "checks-aborted": [
            "suboptimal-compression"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    },
    "color-images": {
        "checks-aborted": [
            "image-depth",
            "resolution-too-high",
            "resolution-too-low",
            "uses-jpeg2000-compression"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    },
```

```
"fonts-results": {
    "checks-aborted": [
        "fontdescriptor-missing-capheight",
        "fontdescriptor-missing-fields",
        "uses-base14fonts-not-embedded",
        "uses-fonts-fully-embedded",
        "uses-fonts-not-embedded"
    ],
    "checks-completed": [ ],
    "errors": { },
    "how-to-optimize": { },
    "information": { }
},
"general-results": {
    "checks-aborted": [
        "born-digital"
        "claims-pdfa-conformance"
        "claims-pdfe-conformance"
        "claims-pdfua-conformance"
        "claims-pdfvt-conformance"
        "claims-pdfx-conformance"
        "contains-owner-password"
        "contains-signature"
        "damaged"
        "image-only"
        "password-protected"
        "pdf-v2"
        "xfa-type"
    ],
    "checks-completed": [
        "unable-to-open"
    ],
    "errors": {
        "unable-to-open": [
            "Cannot be opened/not valid PDF"
        ]
    },
    "how-to-optimize": { },
    "information": { }
},
"grayscale-images": {
```

```
        "checks-aborted": [
            "resolution-too-high",
            "resolution-too-low",
            "uses-jpeg2000-compression"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    },
    "image-results": {
        "checks-aborted": [
            "alternate-images"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    },
    "monochrome-images": {
        "checks-aborted": [
            "resolution-too-high",
            "resolution-too-low",
            "uses-jbig2-compression"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    },
    "objects-results": {
        "checks-aborted": [
            "contains-javascript-actions",
            "contains-thumbnails"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    },
    "userdata-results": {
        "checks-aborted": [
            "contains-annots",
```

```
            "contains-annots-not-for-printing",
            "contains-annots-not-for-viewing",
            "contains-annots-without-normal-appearances",
            "contains-embedded-files",
            "contains-metadata",
            "contains-optional-content",
            "contains-private-data",
            "contains-transparency"
        ],
        "checks-completed": [ ],
        "errors": { },
        "how-to-optimize": { },
        "information": { }
    }
}
```

Setting up your own profile

Your JSON profile file (or files) should include a list of settings that define the kinds of checks you want to apply to your PDF documents. You can make your custom profile file as long or as short as you like.

By default, PDF Checker reports if a PDF cannot be opened for one of the following reasons:

1. PDF is password protected and the password was not provided in the command line statement
2. The PDF document cannot be opened for some other reason.

The message in the report for these conditions can be controlled by the JSON profile. If you don't provide a message in the JSON profile, the PDF Checker software provides a default message to indicate the PDF could not be opened.

Only use lowercase characters for the keys and values you add to the JSON file.

This sample shows settings used to check for fonts that are not embedded in a PDF document:

```
"uses-fonts-not-embedded": {
      "check": "on",
      "report-as-error": "on",
      "report-message": "Uses fonts not embedded in document",
      "abort-remaining-checks": "off"
},
```

Here fonts that are not embedded in the PDF document will be reported as errors. PDF Checker will continue processing other checks after this condition is encountered.

Error Codes

1001	Syntax error. You started PDF Checker and there was an error in your command syntax.
1002	Profile not found. PDF Checker could not find the profile file you named in your command statement. Check the file name or path for errors.
1003	Profile invalid. The JSON file you provided as the profile contains syntax errors.
1004	Invalid flag value. PDF Checker found an unexpected value in the settings in your JSON profile. Key values, such as "uses-fonts-not-embedded" or "XFA-type" as well as the word "on" must always be in quotes.
1005	Results file creation/open failure. Check permissions to create or open the results file you have indicated.
1006	Command line is missing input or JSON file.