7.1.1: February 4, 2018

- Bug fix: files whose /ID fields were other than 16 bytes long can now be properly linearized
- A few compile and link issues have been corrected for some platforms.

7.1.0: January 14, 2018

- PDF files contain streams that may be compressed with various compression algorithms which, in some cases, may be enhanced by various predictor functions. Previously only the PNG up predictor was supported. In this version, all the PNG predictors as well as the TIFF predictor are supported. This increases the range of files that qpdf is able to handle.
- QPDF now allows a raw encryption key to be specified in place of a password when opening encrypted files, and will optionally display the encryption key used by a file. This is a non-standard operation, but it can be useful in certain situations. Please see the discussion of **--password-is-hex-key** in Section 3.3, “Basic Options”, page 4 or the comments around *QPDF::setPasswordIsHexKey* in *QPDF.hh* for additional details.
- Bug fix: numbers ending with a trailing decimal point are now properly recognized as numbers.
- Bug fix: when building qpdf from source on some platforms (especially MacOS), the build could get confused by older versions of qpdf installed on the system. This has been corrected.

7.0.0: September 15, 2017

- Packaging and Distribution Changes
  - QPDF's primary license is now version 2.0 of the Apache License [http://www.apache.org/licenses/LICENSE-2.0] rather than version 2.0 of the Artistic License. You may still, at your option, consider qpdf to be licensed with version 2.0 of the Artistic license.
  - QPDF no longer has a dependency on the PCRE (Perl-Compatible Regular Expression) library. QPDF now has an added dependency on the JPEG library.
- Bug Fixes
  - This release contains many bug fixes for various infinite loops, memory leaks, and other memory errors that could be encountered with specially crafted or otherwise erroneous PDF files.
- New Features
  - QPDF now supports reading and writing streams encoded with JPEG or RunLength encoding. Library API enhancements and command-line options have been added to control this behavior. See command-line options **--compress-streams** and **--decode-level** and methods *QPDFWriter::setCompressStreams* and *QPDFWriter::setDecodeLevel*.
  - QPDF is much better at recovering from broken files. In most cases, qpdf will skip invalid objects and will preserve broken stream data by not attempting to filter broken streams. QPDF is now able to recover or at least not crash on dozens of broken test files I have received over the past few years.
  - Page rotation is now supported and accessible from both the library and the command line.
  - ***QPDFWriter*** supports writing files in a way that preserves PCLm compliance in support of driverless printing. This is very specialized and is only useful to applications that already know how to create PCLm files.
- Enhancements to the **qpdf** Command-line Tool. All new options listed here are documented in more detail in Chapter 3, *Running QPDF*, page 4.