Appendix B. Upgrading from 2.0 to 2.1

Although, as a general rule, we like to avoid introducing source-level incompatibilities in qpdf's interface, there were a few non-compatible changes made in this version. A considerable amount of source code that uses qpdf will probably compile without any changes, but in some cases, you may have to update your code. The changes are enumerated here. There are also some new interfaces; for those, please refer to the header files.

- QPDF's exception handling mechanism now uses ***std::logic_error*** for internal errors and ***std::runtime_error*** for runtime errors in favor of the now removed ***QEXC*** classes used in previous versions. The ***QEXC*** exception classes predated the addition of the *<stdexcept>* header file to the C++ standard library. Most of the exceptions thrown by the qpdf library itself are still of type ***QPDFExc*** which is now derived from ***std::runtime_error***. Programs that caught an instance of ***std::exception*** and displayed it by calling the *what()* method will not need to be changed.
- The ***QPDFExc*** class now internally represents various fields of the error condition and provides interfaces for querying them. Among the fields is a numeric error code that can help applications act differently on (a small number of) different error conditions. See *QPDFExc.hh* for details.
- Warnings can be retrieved from qpdf as instances of ***QPDFExc*** instead of strings.
- The nested ***QPDF::EncryptionData*** class's constructor takes an additional argument. This class is primarily intended to be used by ***QPDFWriter***. There's not really anything useful an end-user application could do with it. It probably shouldn't really be part of the public interface to begin with. Likewise, some of the methods for computing internal encryption dictionary parameters have changed to support `/R=4` encryption.
- The method *QPDF::getUserPassword* has been removed since it didn't do what people would think it did. There are now two new methods: *QPDF::getPaddedUserPassword* and *QPDF::getTrimmedUserPassword*. The first one does what the old *QPDF::getUserPassword* method used to do, which is to return the password with possible binary padding as specified by the PDF specification. The second one returns a human-readable password string.
- The enumerated types that used to be nested in ***QPDFWriter*** have moved to top-level enumerated types and are now defined in the file *qpdf/Constants.h*. This enables them to be shared by both the C and C++ interfaces.