helping with, nor do they attempt to encapsulate underlying structures. They just provide a few methods to help with error-prone, repetitive, or complex tasks. In some cases, a helper object may cache some information that is expensive to gather. In such cases, the helper classes are implemented so that their own methods keep the cache consistent, and the header file will provide a method to invalidate the cache and a description of what kinds of operations would make the cache invalid. If in doubt, you can always discard a helper class and create a new one with the same underlying objects, which will ensure that you have discarded any stale information.

By Convention, document helpers are called ***QPDFSomethingDocumentHelper*** and are derived from ***QPDFDocumentHelper***, and object helpers are called ***QPDFSomethingObjectHelper*** and are derived from ***QPDFObjectHelper***. For details on specific helpers, please see their header files. You can find them by looking at *include/qpdf/QPDF*DocumentHelper.hh* and *include/qpdf/QPDF*ObjectHelper.hh*.

In order to avoid creation of circular dependencies, the following general guidelines are followed with helper classes:

- Core class interfaces do not know about helper classes. For example, no methods of ***QPDF*** or ***QPDFObjectHandle*** will include helper classes in their interfaces.
- Interfaces of object helpers will usually not use document helpers in their interfaces. This is because it is much more useful for document helpers to have methods that return object helpers. Most operations in PDF files start at the document level and go from there to the object level rather than the other way around. It can sometimes be useful to map back from object-level structures to document-level structures. If there is a desire to do this, it will generally be provided by a method in the document helper class.
- Most of the time, object helpers don't know about other object helpers. However, in some cases, one type of object may be a container for another type of object, in which case it may make sense for the outer object to know about the inner object. For example, there are methods in the ***QPDFPageObjectHelper*** that know ***QPDFAnnotationObjectHelper*** because references to annotations are contained in page dictionaries.
- Any helper or core library class may use helpers in their implementations.

Prior to qpdf version 8.1, higher level interfaces were added as “convenience functions” in either ***QPDF*** or ***QPDFObjectHandle***. For compatibility, older convenience functions for operating with pages will remain in those classes even as alternatives are provided in helper classes. Going forward, new higher level interfaces will be provided using helper classes.

7.4. Implementation Notes

This section contains a few notes about QPDF's internal implementation, particularly around what it does when it first processes a file. This section is a bit of a simplification of what it actually does, but it could serve as a starting point to someone trying to understand the implementation. There is nothing in this section that you need to know to use the qpdf library.

QPDFObject is the basic PDF Object class. It is an abstract base class from which are derived classes for each type of PDF object. Clients do not interact with Objects directly but instead interact with ***QPDFObjectHandle***.

When the ***QPDF*** class creates a new object, it dynamically allocates the appropriate type of ***QPDFObject*** and immediately hands the pointer to an instance of ***QPDFObjectHandle***. The parser reads a token from the current file position. If the token is a not either a dictionary or array opener, an object is immediately constructed from the single token and the parser returns. Otherwise, the parser iterates in a special mode in which it accumulates objects until it finds a balancing closer. During this process, the “R” keyword is recognized and an indirect ***QPDFObjectHandle*** may be constructed.

The *QPDF::resolve()* method, which is used to resolve an indirect object, may be invoked from the ***QPDFObjectHandle*** class. It first checks a cache to see whether this object has already been read. If not, it reads the object from the PDF file and caches it. It the returns the resulting ***QPDFObjectHandle***. The calling object handle then replaces