Using LZ4 compression

To use the builtin support for Yann Collet's LZ4 compression, first check that LZ4 is installed in include and library directories searched by the compiler. Once LZ4 is installed, you can enable LZ4 using the `-DENABLE_LZ4=1` option to cmake.

If LZ4 is installed in a location not normally searched by the compiler toolchain, you'll need to modify the include and library paths to indicate these locations. For example, with the LZ4 includes and libraries installed in `/usr/local/include` and `/usr/local/lib`, you would run cmake with the following additional arguments:

```
-DENABLE_LZ4=1 -DCMAKE_INCLUDE_PATH=/usr/local/include -DCMAKE_LIBRARY_PATH=/usr/local/lib
```

When opening the WiredTiger database, load the LZ4 shared library as an extension. For example, with the WiredTiger library installed in `/usr/local/lib`, you would use the following extension:

```
    error_check(wiredtiger_open(
      home, NULL, "create,extensions=[/usr/local/lib/libwiredtiger_lz4.so]", &conn));
```

Finally, when creating the WiredTiger object, set `block_compressor` to `lz4:`

```
    error_check(session->create(
      session, "table:mytable", "block_compressor=lz4,key_format=S,value_format=S"));
```

Using snappy compression

To use the builtin support for Google's snappy compression, first check that snappy is installed in include and library directories searched by the compiler. Once snappy is installed, you can enable snappy using the `-DENABLE_SNAPPY=1` option to cmake.

If snappy is installed in a location not normally searched by the compiler toolchain, you'll need to modify the include and library paths to indicate these locations. For example, with the snappy includes and libraries installed in `/usr/local/include` and `/usr/local/lib`, you would run cmake with the following additional arguments:

```
-DENABLE_SNAPPY=1 -DCMAKE_INCLUDE_PATH=/usr/local/include -DCMAKE_LIBRARY_PATH=/usr/local/lib
```

When opening the WiredTiger database, load the snappy shared library as an extension. For example, with the WiredTiger library installed in `/usr/local/lib`, you would use the following extension:

```
    error_check(wiredtiger_open(
      home, NULL, "create,extensions=[/usr/local/lib/libwiredtiger_snappy.so]", &conn));
```

Finally, when creating the WiredTiger object, set `block_compressor` to `snappy:`

```
    error_check(session->create(
      session, "table:mytable", "block_compressor=snappy,key_format=S,value_format=S"));
```

Using zlib compression

To use the builtin support for Greg Roelofs' and Mark Adler's zlib compression, first check that zlib is installed in include and library directories searched by the compiler. Once zlib is installed, you can enable zlib using the `-DENABLE_ZLIB=1` option to cmake.

If zlib is installed in a location not normally searched by the compiler toolchain, you'll need to modify the include and library paths to indicate these locations. For example, with the zlib includes and libraries installed in `/usr/local/include` and `/usr/local/lib`, you would run cmake with the following additional arguments:

```
-DENABLE_ZLIB=1 -DCMAKE_INCLUDE_PATH=/usr/local/include -DCMAKE_LIBRARY_PATH=/usr/local/lib
```

When opening the WiredTiger database, load the zlib shared library as an extension. For example, with the WiredTiger library installed in `/usr/local/lib`, you would use the following extension:

```
    error_check(wiredtiger_open(
      home, NULL, "create,extensions=[/usr/local/lib/libwiredtiger_zlib.so]", &conn));
```

The default compression level for the zlib compression is `Z_DEFAULT_COMPRESSION` (see the zlib documentation for further information); compression can be configured to other levels using the additional configuration argument `compression_level`.

```
    error_check(wiredtiger_open(home, NULL,
      "create,extensions=[/usr/local/lib/libwiredtiger_zlib.so=[config=[compression_level=3]]]",
      &conn));
```

Finally, when creating the WiredTiger object, set `block_compressor` to `zlib:`

```
    error_check(session->create(
      session, "table:mytable", "block_compressor=zlib,key_format=S,value_format=S"));
```

Using Zstd compression

To use the builtin support for Facebook's Zstd compression, first check that Zstd is installed in include and library directories searched by the compiler. Once Zstd is installed, you can enable Zstd using the `-DENABLE_ZSTD=1` option to cmake.

If Zstd is installed in a location not normally searched by the compiler toolchain, you'll need to modify the include and library paths to indicate these locations. For example, with the Zstd includes and libraries installed in `/usr/local/include` and `/usr/local/lib`, you would run cmake with the following additional arguments:

```
-DENABLE_ZSTD=1 -DCMAKE_INCLUDE_PATH=/usr/local/include -DCMAKE_LIBRARY_PATH=/usr/local/lib
```