API

The wiki pages are not used for documentation any more. Please visit http://bookkeeper.apache.org/docs/latest/api/overview/ for latest API documentation.

(this page will contain a description of all the Client API of BookKeeper)

Introduction to the Client API

BookKeeper Client API entry point is the org.apache.bookkeeper.client.BookKeeper class.

Usually a Client creates an instance of BookKeeper and uses it to manage several Ledgers using the LedgerHandle API.

```
try (BookKeeper bookkeeper = BookKeeper.forConfig(new ClientConfiguration()).build();
       LedgerHandle ledger = bookkeeper.createLedger(BookKeeper.DigestType.CRC32, "foo".getBytes())) {
       ledger.addEntry("bar".getBytes());

}
```

BookKeeper Client Builder API

In order to create a BookKeeper client you have to setup a ClientConfiguration object.

The most important parameter is the ZooKeeper connection string, which is needed to access to the BookKeeper ensemble.

```
ClientConfiguration config = new ClientConfiguration();
config.setZkServers("localhost:1281");
try (BookKeeper bookKeeper = BookKeeper.forConfig(config).build();) {
      // ...
}
```

If you already have a ZooKeeper handle and you want to share it you have to pass it the Builder

```
ZooKeeper zookeeper = ....;
ClientConfiguration config = new ClientConfiguration();
config.setZkServers("localhost:1281");
try (BookKeeper bookKeeper = BookKeeper
        .forConfig(config)
        .setZookeeper(zookeeper)
        .build();) {
      // ...
}
```

Beware that ClientConfiguration will be automatically filled with System.getProperties()

Synch vs Asynch

BookKeeper API methods provide access to BookKeeper features, internally all is asynchronous and in general the asynch API will give you more control about whats going on and result in better performances.

For instance

```
try (LedgerHandle handle = bookkeeper.createLedger(1, 1, 1, BookKeeper.DigestType.CRC32, "test".getBytes())) {
     byte[] data = "test".getBytes();
     final CountDownLatch la = new CountDownLatch(SIZE);
     for (int i = 0; i < SIZE; i++) {
         handle.asyncAddEntry(data, new AsyncCallback.AddCallback() {
                 @Override
            public void addComplete(int rc, LedgerHandle lh, long l, Object o) {
                            if (rc != BKException.Code.OK) {
                     System.err.println("Error: "+BKException.getMessage(rc));
                }
                assertEquals("callback_param",o);
                la.countDown();
            }
         }, "callback_param");
     }
     la.await();
}
```

A similar code using the synch API is:

```
try (LedgerHandle handle = bookkeeper.createLedger(1, 1, 1, BookKeeper.DigestType.CRC32, "test".getBytes())) {
     byte[] data = "test".getBytes();
     for (int i = 0; i < SIZE; i++) {
         handle.addEntry(data);
     }
}
```

The async API will be really more efficient from the standpoint of speed but you have to deal with callbacks, remember that:

- callbacks are called in a different thread
- exception will not be thrown in the main (callee) thread but you have to switch over the 'rc' (return code) parameter, using the BKException.Code class or BKException.create(rc) method
- you have no direct control on the pool of threads which executes the callbacks

Writing to a Ledger

Reading from a Ledger

(brief explanation of fencing)

Closing a Ledger

You always have to call LedgerHandle.close(), this operation will ensure the consistency of all metadata of the ledger and will release resources assigned to the ledger, both locally to the client and on Bookies.

Deleting a Ledger

When a Ledger is no more in use it is better to delete it, in order to let all the Bookies release all the resources retained by the ledger. A ledger will also retain resources on ZooKeeper.

The code is simple:

```
bookkeeper.deleteLedger(id);
```

List ledgers

If you need to discover all the ledgersyou can use the BookKeeperAdmin tool

```
BookKeeperAdmin admin = new BookKeeperAdmin(bookkeeper);
for (long id : admin.listLedgers()) {
    LedgerHandle h = admin.openLedgerNoRecovery(id);
    LedgerMetadata meta = admin.getLedgerMetadata(h);
}
```

Return Codes