```
////////////// API  ////////////

// Captures all Hive connection info


public class HiveStreamingEndPoint {
  public final String nameNodeUri;
  public final String metaStoreUri;
  public final String database;
  public final String table;
  public final List<String> partitionVals;

  public HiveStreamingEndPoint(String nameNodeUri, String metaStoreUri
                               , String database, String table
                               , List<String> partitionVals);

  // Connects to the end point and returns the connection object
  public StreamingConnection newConnection(String serdeClassName, boolean
useEmbeddedMetastore);
}
```

```
// The Streaming connection – for acquiring Transaction Batches

public interface StreamingConnection {
   // Acquire a set of transactions
   public TransactionBatch fetchTransactionBatch(int numTransactionsHint)
           throws ConnectionError, InvalidPartition, StreamingException;

  // Close connection
   public void close();
}
```

***** fetchTransactionBatch(int numTransactions) throws ...**
Acquires a new batch of transactions from Hive.
numTransactions is a hint from client indicating how many transactions
client needs

***** close()**
Close any open connections and resources.

```
public interface TransactionBatch  {
  public enum TxnState {INACTIVE, OPEN, COMMITTED, ABORTED }

  public void beginNextTransaction() throws StreamingException;
```

```
  public TxnState getCurrentTransactionState();
  public void commit() throws StreamingException;
  public void abort() throws StreamingException;

  public int remainingTransactions();


  // Write Data for current Txn //
  public void write(byte[] record) throws ConnectionError, IOException,
StreamingException;
  public void write(Collection<byte[]> records) throws ConnectionError,
IOException, StreamingException;

  // Close batch
  public void close();
}
```

***** beginNextTransaction(...) throws ...**
Switch to the next transaction in the batch.
returns false if there are no more transactions.

***** commit()**
Commits the currently open transaction.

***** abort() throws ...**
Aborts the currently open transaction.

***** write(byte[] record)**
Write a record.

***** write(Collection<byte[]> records)**
Write multiple records

***** getTransactionState()**
Get the current state of the transaction

***** remainingTransactions()**
Get a count of the unused transactions in the batch acquired by the last call to fetchTransactionBatch. Current transaction is not considered part of remaining transactions.

***** close()**
Close the transacton batch

```
///// Example – stream five records in two transactions /////


// Assumed HIVE table Schema:
// partitions(continent: string, country: string)
// columns(id: int, msg: string)

String dbName = "testing";
String tblName = "alerts";
ArrayList<String> partitionVals = new ArrayList<String>(2);
partitionVals.add("Asia");
partitionVals.add("India");
String serdeClass = "org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe";


HiveStreamingEndPoint hiveEP = new HiveStreamingEndPoint(
"localhost://namnode.uri", null /* embedded metastore */, dbName, tblName,
partitionVals);
StreamingConnection connection = hiveEP.newConnection(serdeClass, true);
TransactionBatch txnBatch =  connection.fetchTransactionBatch(10);

///// First TXN
txnBatch.beginNextTransaction();
txnBatch.write("1^AHello streaming".getBytes());
txnBatch.write("2^AWelcome to streaming".getBytes());
txnBatch.commit();

///// Second TXN
txnBatch.beginNextTransaction();
txnBatch.write("3^ARoshan Naik".getBytes());
txnBatch.write("4^AAlan Gates".getBytes());
txnBatch.write("5^AOwen O'Malley".getBytes());
txnBatch.commit();

connection.close();
```