#include "internal/endianutils.h"

#include "internal/error.h"

#include "internal/thrift.h"

#include <twml/ThriftWriter.h>

#include <twml/Error.h>

#include <twml/io/IOError.h>

#include <cstring>

using namespace twml::io;

namespace twml {

template <typename T> inline

uint64\_t ThriftWriter::write(T val) {

if (!m\_dry\_run) {

if (m\_bytes\_written + sizeof(T) > m\_buffer\_size)

throw IOError(IOError::DESTINATION\_LARGER\_THAN\_CAPACITY);

memcpy(m\_buffer, &val, sizeof(T));

m\_buffer += sizeof(T);

}

m\_bytes\_written += sizeof(T);

return sizeof(T);

}

TWMLAPI uint64\_t ThriftWriter::getBytesWritten() {

return m\_bytes\_written;

}

TWMLAPI uint64\_t ThriftWriter::writeStructFieldHeader(int8\_t field\_type, int16\_t field\_id) {

return writeInt8(field\_type) + writeInt16(field\_id);

}

TWMLAPI uint64\_t ThriftWriter::writeStructStop() {

return writeInt8(static\_cast<int8\_t>(TTYPE\_STOP));

}

TWMLAPI uint64\_t ThriftWriter::writeListHeader(int8\_t element\_type, int32\_t num\_elems) {

return writeInt8(element\_type) + writeInt32(num\_elems);

}

TWMLAPI uint64\_t ThriftWriter::writeMapHeader(int8\_t key\_type, int8\_t val\_type, int32\_t num\_elems) {

return writeInt8(key\_type) + writeInt8(val\_type) + writeInt32(num\_elems);

}

TWMLAPI uint64\_t ThriftWriter::writeDouble(double val) {

int64\_t bin\_value;

memcpy(&bin\_value, &val, sizeof(int64\_t));

return writeInt64(bin\_value);

}

TWMLAPI uint64\_t ThriftWriter::writeInt8(int8\_t val) {

return write(val);

}

TWMLAPI uint64\_t ThriftWriter::writeInt16(int16\_t val) {

return write(betoh16(val));

}

TWMLAPI uint64\_t ThriftWriter::writeInt32(int32\_t val) {

return write(betoh32(val));

}

TWMLAPI uint64\_t ThriftWriter::writeInt64(int64\_t val) {

return write(betoh64(val));

}

TWMLAPI uint64\_t ThriftWriter::writeBinary(const uint8\_t \*bytes, int32\_t num\_bytes) {

writeInt32(num\_bytes);

if (!m\_dry\_run) {

if (m\_bytes\_written + num\_bytes > m\_buffer\_size)

throw IOError(IOError::DESTINATION\_LARGER\_THAN\_CAPACITY);

memcpy(m\_buffer, bytes, num\_bytes);

m\_buffer += num\_bytes;

}

m\_bytes\_written += num\_bytes;

return 4 + num\_bytes;

}

TWMLAPI uint64\_t ThriftWriter::writeString(std::string str) {

return writeBinary(reinterpret\_cast<const uint8\_t \*>(str.data()), str.length());

}

TWMLAPI uint64\_t ThriftWriter::writeBool(bool val) {

return write(val);

}

} // namespace twml