public abstract class FrontendConnection extends AbstractConnection {

protected long id;

protected String host;

protected int port;

protected int localPort;

protected long idleTimeout;

protected byte[] seed;

protected String user;

protected String schema;

protected String executeSql;

protected FrontendPrivileges privileges;

protected FrontendQueryHandler queryHandler;

protected FrontendPrepareHandler prepareHandler;

protected LoadDataInfileHandler loadDataInfileHandler;

protected boolean isAccepted;

protected boolean isAuthenticated;

public FrontendConnection(NetworkChannel channel) throws IOException {

super(channel);

InetSocketAddress localAddr = (InetSocketAddress) channel.getLocalAddress();

InetSocketAddress remoteAddr = null;

if (channel instanceof SocketChannel) {

remoteAddr = (InetSocketAddress) ((SocketChannel) channel).getRemoteAddress();

} else if (channel instanceof AsynchronousSocketChannel) {

remoteAddr = (InetSocketAddress) ((AsynchronousSocketChannel) channel).getRemoteAddress();

}

this.host = remoteAddr.getHostString();

this.port = localAddr.getPort();

this.localPort = remoteAddr.getPort();

this.handler = new FrontendAuthenticator(this);

}

public FrontendConnection() throws IOException {

super();

}

public long getId()

public void setId(long id)

public String getHost()

public void setHost(String host)

public int getPort()

public void setPort(int port)

public int getLocalPort()

public void setLocalPort(int localPort)

public void setAccepted(boolean isAccepted)

public void setProcessor(NIOProcessor processor)

public LoadDataInfileHandler getLoadDataInfileHandler()

public void setLoadDataInfileHandler(LoadDataInfileHandler loadDataInfileHandler)

public void setQueryHandler(FrontendQueryHandler queryHandler)

public void setPrepareHandler(FrontendPrepareHandler prepareHandler)

public void setAuthenticated(boolean isAuthenticated)

public FrontendPrivileges getPrivileges()

public void setPrivileges(FrontendPrivileges privileges)

public String getUser()

public void setUser(String user)

public String getSchema()

public void setSchema(String schema)

public String getExecuteSql()

public void setExecuteSql(String executeSql)

public byte[] getSeed()

public boolean setCharsetIndex(int ci)

public void writeErrMessage(int errno, String msg)

public void writeErrMessage(byte id, int errno, String msg)

public void initDB(byte[] data)

public void loadDataInfileStart(String sql)

public void loadDataInfileData(byte[] data)

public void loadDataInfileEnd(byte packID)

public void query(String sql)

public void query(byte[] data)

public void stmtPrepare(byte[] data)

public void stmtSendLongData(byte[] data)

public void stmtReset(byte[] data)

public void stmtExecute(byte[] data)

public void stmtClose(byte[] data)

public void ping()

public void heartbeat(byte[] data)

public void kill(byte[] data)

public void unknown(byte[] data)

public void register() throws IOException

public void handle(final byte[] data)

public void rawHandle(final byte[] data)

protected int getServerCapabilities()

protected boolean isConnectionReset(Throwable t)

public String toString()

private final static byte[] encodeString(String src, String charset)

public void close(String reason)

}