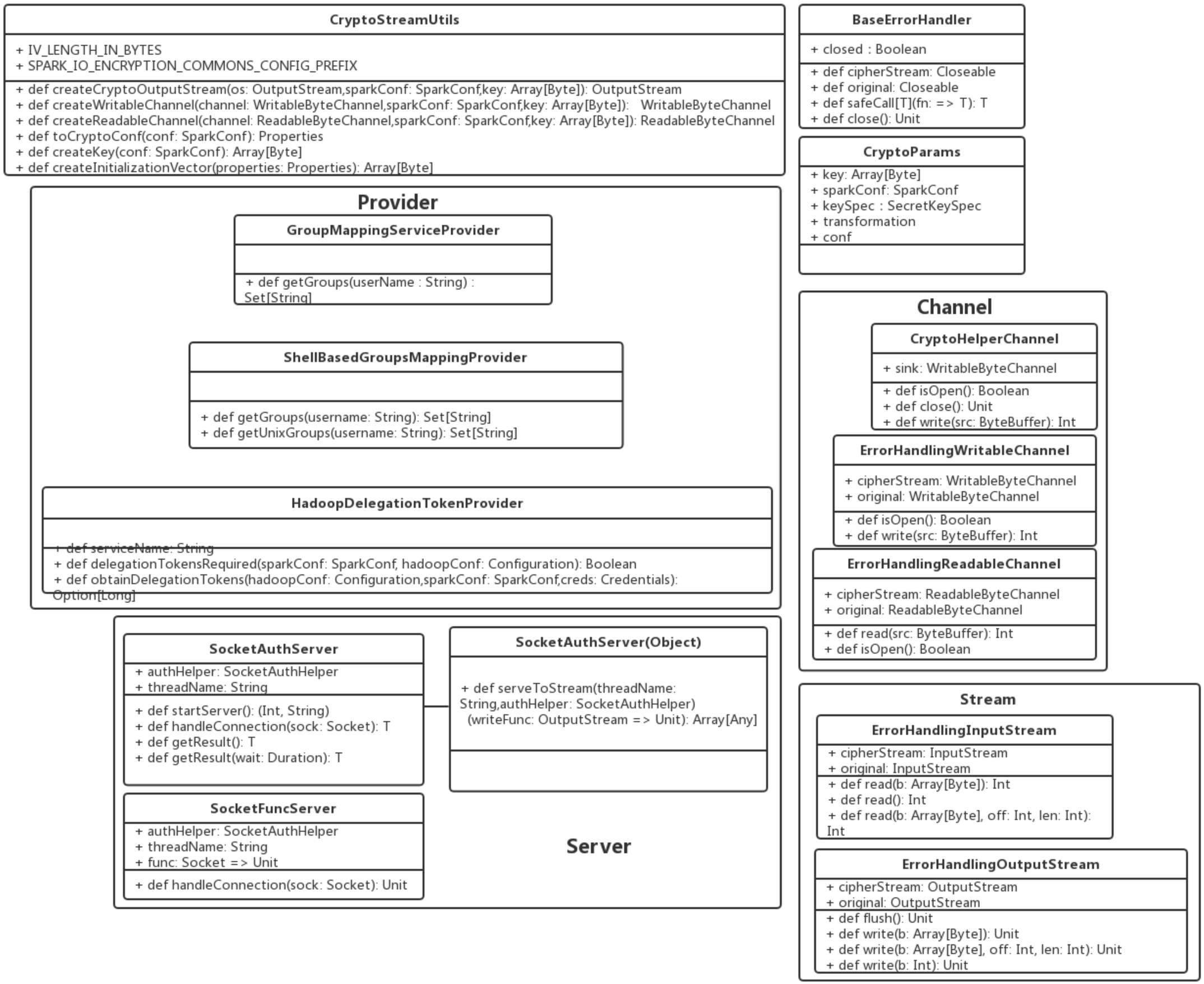
CryptoStreamUtils
+ IV_LENGTH_IN_BYTES
+ SPARK_IO_ENCRYPTION_COMMONS_CONFIG_PREFIX
+ def createCryptoOutputStream(os: OutputStream,sparkConf: SparkConf,key: Array[Byte]): OutputStream
+ def createWritableChannel(channel: WritableByteChannel,sparkConf: SparkConf,key: Array[Byte]): WritableByteChannel
+ def createReadableChannel(channel: ReadableByteChannel,sparkConf: SparkConf,key: Array[Byte]): ReadableByteChannel
+ def toCryptoConf(conf: SparkConf): Properties
+ def createKey(conf: SparkConf): Array[Byte]
+ def createInitializationVector(properties: Properties): Array[Byte]
BaseErrorHandler
+ closed : Boolean
+ def cipherStream: Closeable
+ def original: Closeable
+ def safeCall[T](fn: => T): T
+ def close(): Unit
CryptoParams
+ key: Array[Byte]
+ sparkConf: SparkConf
+ keySpec : SecretKeySpec
+ transformation
+ conf
Provider
GroupMappingServiceProvider
+ def getGroups(userName : String) : Set[String]
ShellBasedGroupsMappingProvider
+ def getGroups(username: String): Set[String]
+ def getUnixGroups(username: String): Set[String]
HadoopDelegationTokenProvider
+ def serviceName: String
+ def delegationTokensRequired(sparkConf: SparkConf, hadoopConf: Configuration): Boolean
+ def obtainDelegationTokens(hadoopConf: Configuration,sparkConf: SparkConf,creds: Credentials): Option[Long]
Channel
CryptoHelperChannel
+ sink: WritableByteChannel
+ def isOpen(): Boolean
+ def close(): Unit
+ def write(src: ByteBuffer): Int
ErrorHandlingWritableChannel
+ cipherStream: WritableByteChannel
+ original: WritableByteChannel
+ def isOpen(): Boolean
+ def write(src: ByteBuffer): Int
ErrorHandlingReadableChannel
+ cipherStream: ReadableByteChannel
+ original: ReadableByteChannel
+ def read(src: ByteBuffer): Int
+ def isOpen(): Boolean
SocketAuthServer
+ authHelper: SocketAuthHelper
+ threadName: String
+ def startServer(): (Int, String)
+ def handleConnection(sock: Socket): T
+ def getResult(): T
+ def getResult(wait: Duration): T
SocketAuthServer(Object)
+ def serveToStream(threadName: String,authHelper: SocketAuthHelper) (writeFunc: OutputStream => Unit): Array[Any]
SocketFuncServer
+ authHelper: SocketAuthHelper
+ threadName: String
+ func: Socket => Unit
+ def handleConnection(sock: Socket): Unit
Server
Stream
ErrorHandlingInputStream
+ cipherStream: InputStream
+ original: InputStream
+ def read(b: Array[Byte]): Int
+ def read(): Int
+ def read(b: Array[Byte], off: Int, len: Int): Int
ErrorHandlingOutputStream
+ cipherStream: OutputStream
+ original: OutputStream
+ def flush(): Unit
+ def write(b: Array[Byte]): Unit
+ def write(b: Array[Byte], off: Int, len: Int): Unit
+ def write(b: Int): Unit