

ApplicationFinished

Master-->Worker

+ id: String

KillDriver

+ driverId: String

+ def restUri: Option[String

+ executors: List[ExecutorRunner]

+ finishedDrivers: List[DriverRunner]

+ drivers: List[DriverRunner]

+ memory: Int

+ coresUsed: Int

+ memoryUsed: Int

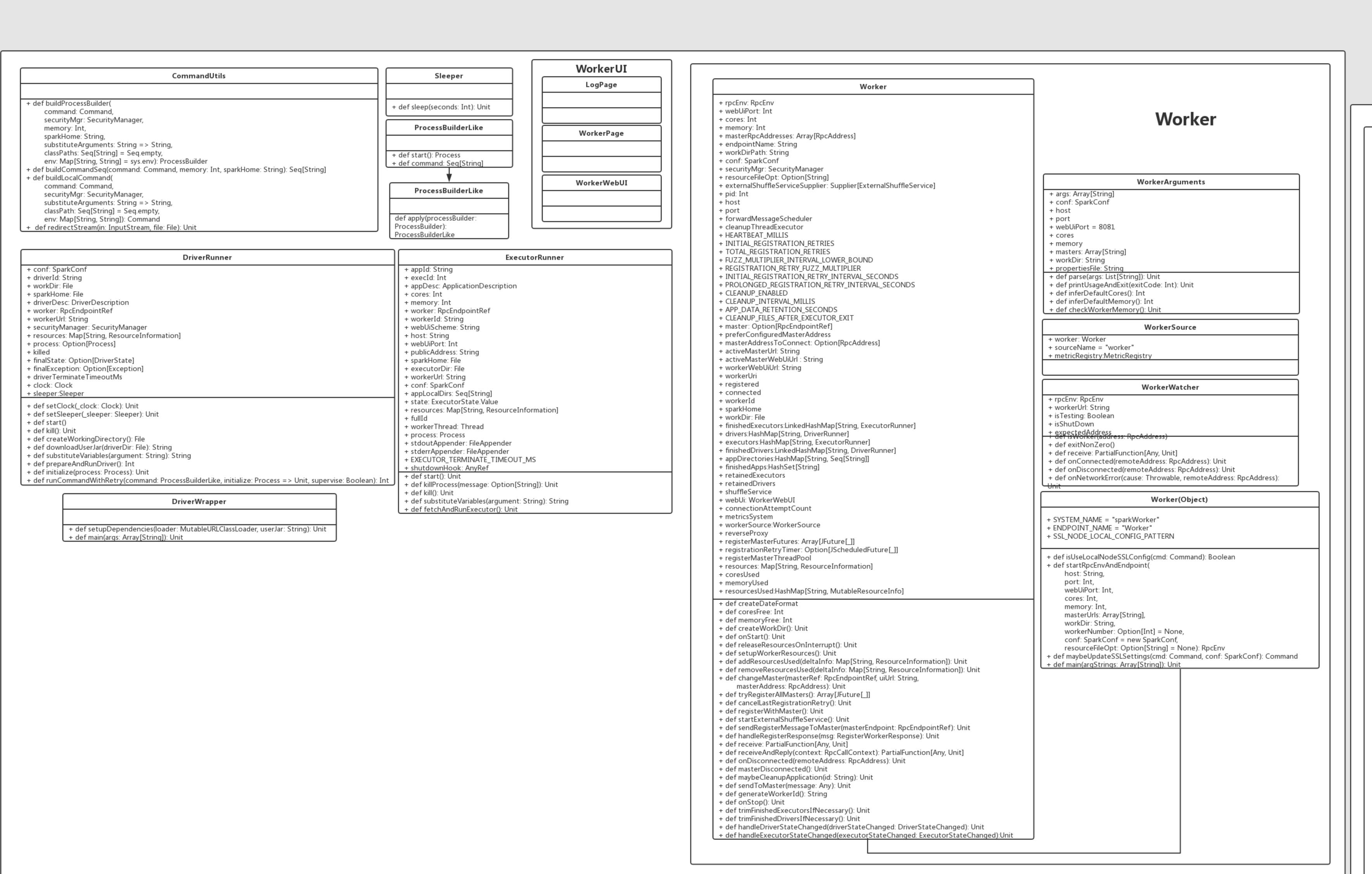
+ masterWebUiUrl: String

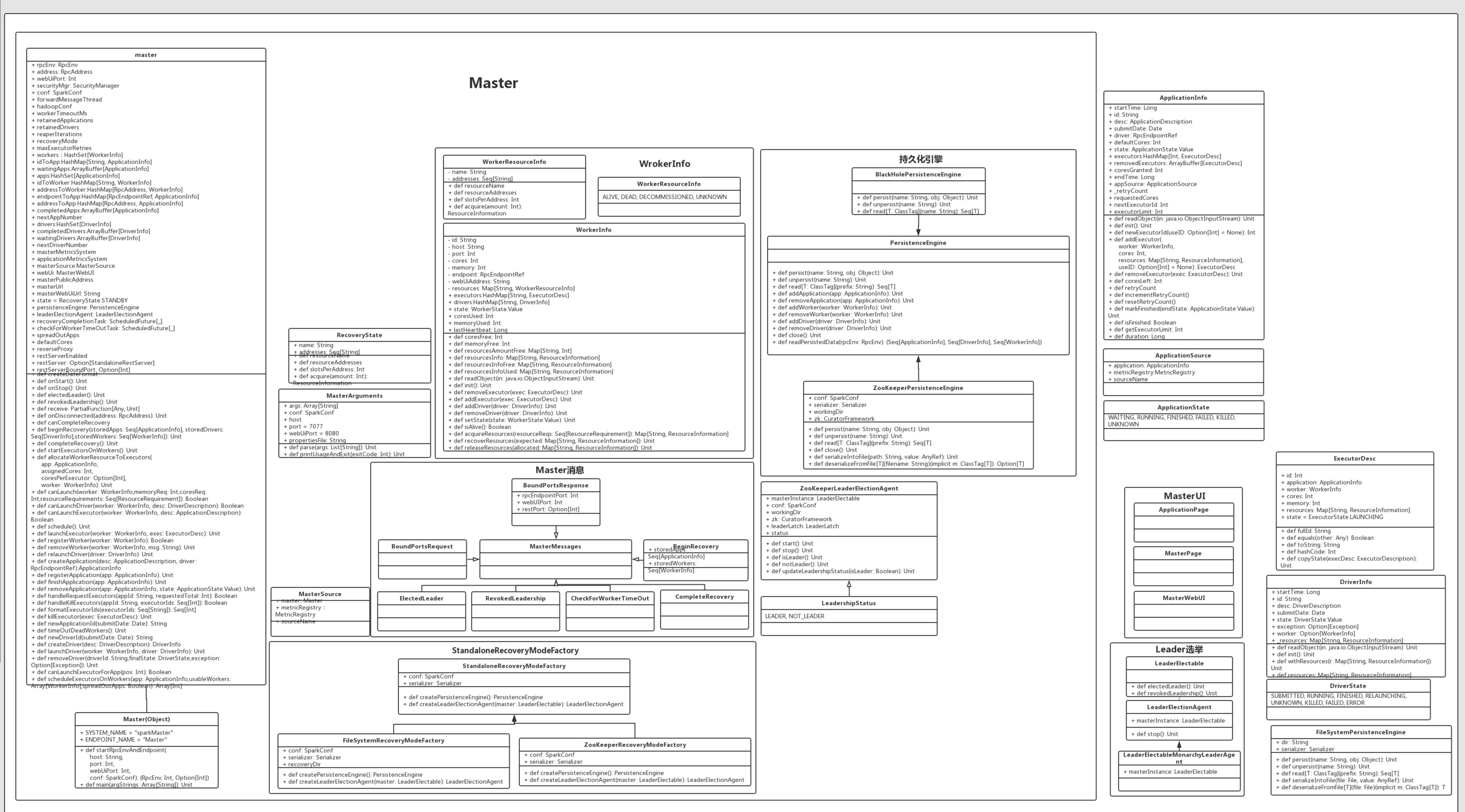
+ finishedExecutors: List[ExecutorRunner]

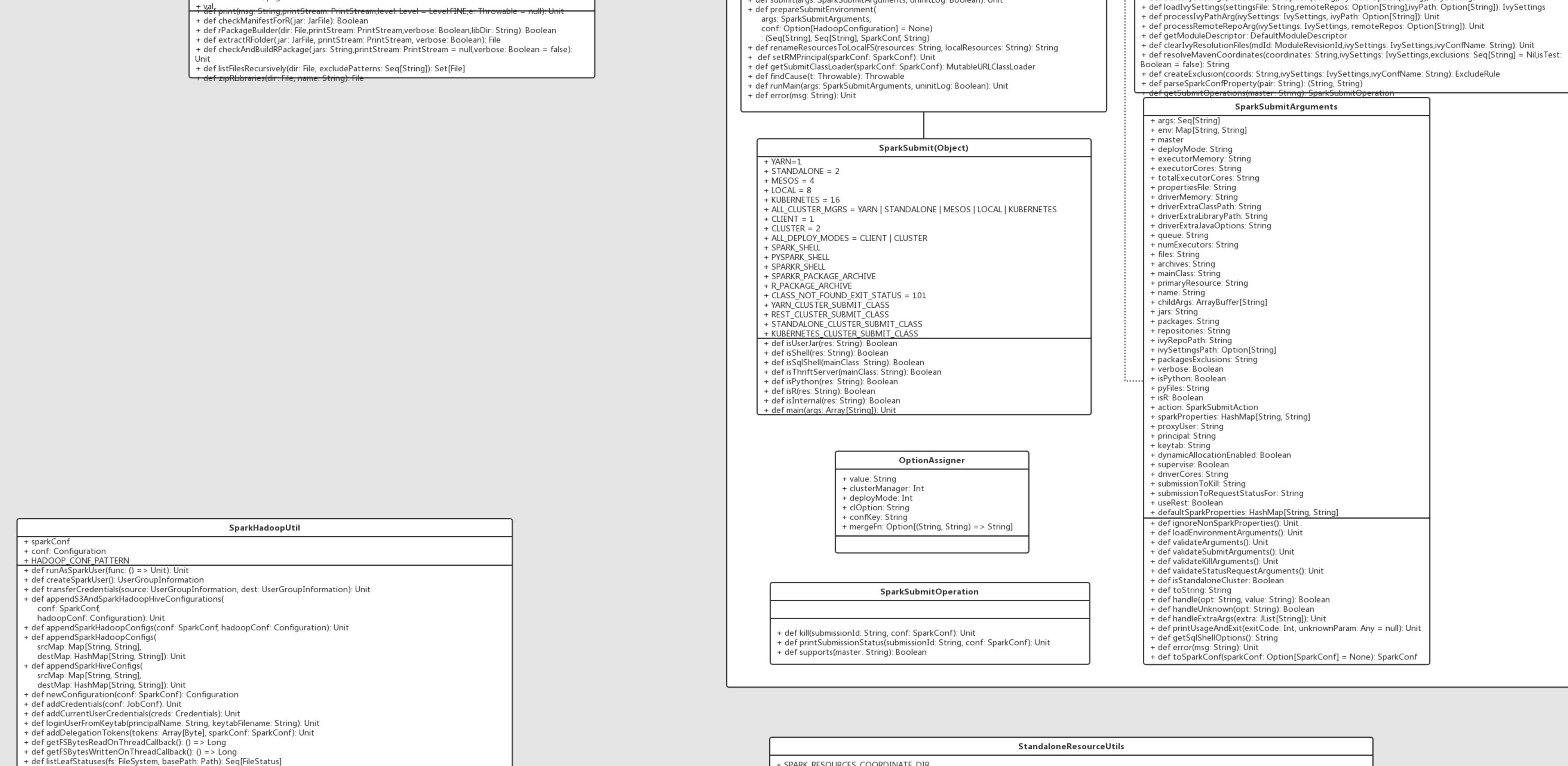
+ resources: Map[String, ResourceInformation]

+ resourcesUsed: Map[String, ResourceInformation]

WorkerStateResponse







SparkDocker (Object)

TestWorkerInfo

DockerId

+ def toString: String

SUBMIT, KILL, REQUEST_STATUS, PRINT_VERSION

+ attribute1:type = defaultValue

+ def printVersion(): Unit

+ def doRunMain(): Unit

+ def doSubmit(args: Array[String]): Unit

+ def kill(args: SparkSubmitArguments): Unit

+ def requestStatus(args: SparkSubmitArguments): Unit

+ def parseArguments(args: Array[String]): SparkSubmitArguments

+ def submit(args: SparkSubmitArguments, uninitLog: Boolean): Unit

SparkSubmit

SparkCuratorUtil

ZK CONNECTION TIMEOUT MILLIS

+ def mkdir(zk: CuratorFramework, path: String): Unit

+ def deleteRecursive(zk: CuratorFramework, path: String):

ZK SESSION TIMEOUT_MILLIS

+ MAX_RECONNECT_ATTEMPTS

RETRY WAIT MILLIS

+ conf: Configuration

conf: SparkConf,

+ def createSparkUser(): UserGroupInformation

hadoopConf: Configuration): Unit

destMap: HashMap[String, String]): Unit

destMap: HashMap[String, String]): Unit

+ def recurse(status: FileStatus): Seq[FileStatus]

+ def globPath(fs: FileSystem, pattern: Path): Seq[Path]

+ def globPathIfNecessary(pattern: Path): Seq[Path]

exclusionSuffix: String): Array[FileStatus]

+ def serialize(creds: Credentials): Array[Byte]

+ def isGlobPath(pattern: Path): Boolean

+ def listFilesSorted(

remoteFs: FileSystem,

+ def listLeafStatuses(fs: FileSystem, baseStatus: FileStatus): Seq[FileStatus]

+ def listLeafDirStatuses(fs: FileSystem, baseStatus: FileStatus): Seq[FileStatus]

+ def substituteHadoopVariables(text: String, hadoopConf: Configuration): String

SparkHadoopUtil(Object)

+ def appendS3AndSparkHadoopHiveConfigurations(conf: SparkConf,hadoopConf: Configuration): Unit

+ def appendSparkHadoopConfigs(conf: SparkConf, hadoopConf: Configuration): Uni

+ def appendSparkHiveConfigs(conf: SparkConf, hadoopConf: Configuration): Unit

+ def createFile(fs: FileSystem, path: Path, allowEC: Boolean): FSDataOutputStream

+ def listLeafDirStatuses(fs: FileSystem, basePath: Path): Seq[FileStatus]

+ def globPathIfNecessary(fs: FileSystem, pattern: Path): Seq[Path]

+ def getSuffixForCredentialsPath(credentialsPath: Path): Int

+ def tokenToString(token: Token[_ <: TokenIdentifier]): String

+ SPARK YARN CREDS TEMP EXTENSION

+ UPDATE_INPUT_METRICS_INTERVAL_RECORDS

+ def newConfiguration(conf: SparkConf): Configuration

+ SPARK YARN CREDS COUNTER DELIM

+ def dumpTokens(credentials: Credentials): Iterable[String]

+ def deserialize(tokenBytes: Array[Byte]): Credentials

+ def isProxyUser(ugi: UserGroupInformation): Boole

+ def get: SparkHadoopUtil

+ def addCredentials(conf: JobConf): Unit

+ def appendSparkHadoopConfigs(

+ def appendSparkHiveConfigs(

srcMap: Map[String, String],

+ SPARK RESOURCES COORDINATE DIR + ALLOCATED RESOURCES FIL + def acquireResources(conf: SparkConf,componentName: String,resources: Map[String, ResourceInformation], pid: Int): Map[String, ResourceInformation] + def releaseResources(conf: SparkConf,componentName: String,toRelease: Map[String, ResourceInformation],pid: Int): Unit + def acquireLock(conf: SparkConf): FileLock + def releaseLock(lock: FileLock): Unit + def allocatedStandaloneResources(resourcesFile: String): Seq[StandaloneResourceAllocation] + def prepareResourcesFile(componentName: String,resources: Map[String, ResourceInformation],dir: File): Option[File] + def writeResourceAllocationJson[T](componentName: String,allocations: Seq[T],jsonFile: File): Unit def toMutable(immutableResources: Map[String, ResourceInformation]): Map[String, MutableResourceInfo] + def formatResourcesDetails(usedInfo: Map[String, ResourceInformation],freeInfo: Map[String, ResourceInformation]): String + def formatResourcesUsed(resourcesTotal: Map[String, ResourceInformation],resourcesUsed: Map[String, ResourceInformation]): String

In Process Spark Submit

- def main(args: Array[String]): Unit

+ def m2Path: File

+ def createRepoResolvers(defaultIvvUserDir: File): ChainResolver

+ def resolveDependencyPaths(artifacts: Array[AnyRef],cacheDirectory: File): String

+ def buildIvySettings(remoteRepos: Option[String],ivyPath: Option[String]): IvySettings

coordinates: String): Seq[MavenCoordinate]

+ def addDependenciesToIvy(md: DefaultModuleDescriptor,artifacts: Seq[MavenCoordinate],ivyConfName: String):

+ def addExclusionRules(ivySettings: IvySettings,ivyConfName: String,md: DefaultModuleDescriptor): Unit