# 日志系统设计

## 框架设计

1. 每个进程一个Log文件(SQLITE)，子进程和线程共享该Log文件。
2. 通常情况下由SCADA和FES两个主进程建立LOG服务，基于主进程的应用，或者子模块调用log接口即可输出log。

2、主进程负责log的创建和关闭;子进程在使用日志接口写日志前，需打开日志文件，退出时关闭日志文件；线程可以直接写日志。

3、写日志接口需传入模块名称，日志和日志等级。在接口内部，需获取当前时间（包含毫秒），进程号、线程号信息

4、日志输出

本地建立SOCKET服务端，日志可以输出至所有连接的trace客户端进程；

Trace进程可以根据进程名，模块名和日志级别对日志进行过滤显示;

本地建立socket客户端，主动连接到dbgstudio（服务端），并且发送log数据。

5、线程和子进程将日志发往主进程的共享内存日志队列，主进程从该队列读出日志，写入本地缓存队列，主进程从本地缓存队列读出日志，写日志，并发至trace

6、LOG 定时读取 bin下面的log.xml文件，查看配置，配置中可以对log的运行状态进行配置。

7、LOG文件的存储策略，如果是当前可执行文件是在根目录下面，则log存储当前目录下的子目录log，如果不是，则在上一级的log目录。

8、LOG的文件名： 进程名称+序号.log，log文件总共有3个（可配置），轮流写，如果当前log的写满了，则自动切换到新的文件中，如果都写满，则切换到第一个，此时，原文件则被覆盖。

注意：LOG需要注意SCADA与FES运行在同一进程空间内的情况！！！（目前暂不支持）

## 二、接口设计

1、主程序创建log：

CreateLog(const char\* szLogName);

* 1. 创建日志文件
  2. 创建与子进程通讯的消息队列

2、主程序关闭log：

DestroyLog(const char\* szLogName);

2.1 关闭日志文件

2.2 销毁与子进程通讯的消息队列

3、子进程打开Log

OpenLog(const char \*szLogName)

打开与主进程通讯的消息队列

4、子进程关闭Log

CloseLog(const char \*szLogName)

关闭与主进程通讯的消息队列

1. 写日志：(非阻塞)

DLog(char \* pszMoudleName,char \*szLogTxt,int nLevel);

将日志写入主进程的共享内存队列