1. **从pspace中读取实时报警数据并以post请求推送到restful接口。**
   1. 经测试pspace接口支持一次读取报警条数在5000～5010之间，报警条数过多会导致读数失败。
   2. bin/release/config.ini可以对pspace链接参数和restful接口的url进行配置。
2. **bin/release/log\_conf.properties工程调试时，可以对log信息进行配置。**
   1. log信息只写文件、log信息只在命令行打印、写文件和打印同时进行、log输出等级配置(error、warning、information)、log文件路径配置......
3. **程序的可扩展性：**
   1. 采用多线程，一个小任务是一个线程(读报警数据和推送数据各是一个线程)，后续可实现同时将从pspace读到的报警数据推送至restful接口和关系数据库。
4. **后续可能要实现的功能：**
   1. 在配置文件中配置需要PUSH的报警等级。
   2. 配置文件能够动态加载，改变配置文件后能及时生效。
   3. 根据从Restful接口请求到的报警等级和报警类型进行PUSH。
   4. 根据配置点表筛选需要PUSH报警的点（不选择点的话推送全部报警，选与不选在config.ini中体现）。
5. **开发过程中遇到的问题。**
   1. httpClient PUSH数据异常或者httpServer response数据异常能捕获到，但导致异常的具体原因不清楚。
   2. 读取速度快，推送速度慢，待databuf达到一定长度后将其清空。
      1. 每次推送的是databuf中先存入的数据，如果databuf长度过长，会影响实时报警的性能。
      2. databuf中每次只存储最新查询到的报警数据，如果因为网络等原因导致推送不及时，会造成报警数据的丢失。
   3. Log类中，将log对象存在类的成员变量中。
   4. Log的过滤等级为trace时，trace及debug等级的内容找不出来。
   5. Databuf中的一个元素代表一条报警数据，不应该是一条元素代表许多条报警数据。
   6. 直接点击exe执行时，读取配置文件失败(修改文件中配置文件的路径),配置项目Debug属性下的工作目录为$(Target/Dir).
   7. 多线程在利用共享队列的时候尽量去进行共享队列的拷贝。
   8. 看下代码规范，注意下其中的变量定义，自己定义的变量使用m\_；
   9. 利用boost中的sleep，尽量不要利用windows的系统函数，这不利于跨平台的使用。
   10. new的地方注意去释放。
   11. 日志模块利用log4。
   12. 利用boost库中的线程。
   13. 做预留接口，在配置文件中配置是使用restful接口还是关系数据库接口，这样的话有利于程序以后的扩展。
   14. 与restful接口处的崩溃主要出现在字符格式的转换，post请求处做字符格式的测试。
   15. 看下同步工具的地方是怎样转到各种关系库的。
   16. 以后做定制的话需要写需求文档、概要设计、详细设计。
   17. 项目的版本号的问题，参考下wiki文档。
   18. ini文件的注释和读取。
   19. 在psot请求推送数据的时候对post请求的超时时间做个设置。