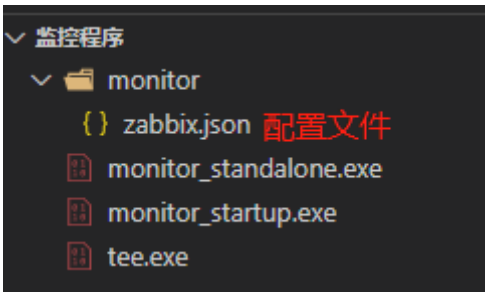
监控对接实施方案

1. 对接的程序

监控对接的程序包括目前运维需要监控的各个应用程序

程序类型	接入方式	备注
壳应用程序	使用 monitor.exe启动 此类应用程序	eg: 通讯壳程序
独立启动的应用程序	使用 monitor.exe启动 此类应用程序	eg: 新交所一级行情
由壳程序启动的应用程序	由壳程序启动后， 再启用 monitor_standalone.exe	eg: CME通讯程序，外挂IB通讯程序

2. 监控程序的使用



其中 monitor/zabbix.json 为配置文件

```
{
  "host": "host1",     // zabbix 监控中 hostname
  "key": "market_L1_sgx",      // 对应 zabbix 监控中 该应用程序 监控项的 key
  "zabbix_server": "180.76.115.130",    // zabbix server 地址
  "zabbix_port": 10051,                 // zabbix server 端口
  "zabbix_submit_rule": "MONITORFLAG", // 应用程序输出内容 提交上报的 过滤选项， 可使用正则表达式
  "monitor_exe": "learning.exe",        // 需要监控 的应用程序的 路径名称
  "monitor_lineNum": 0,                 // 暂未启用
  "monitor_out": "monitor/stdout.log", //  若${monitor_out_flag} 配置为 1, monitor_out应 配置为 程序监控内容输出的文件名，适用于标准输出作为监控内容的程序；  若${monitor_out_flag} 配置为 0, monitor_out应配置为 应用程序自行规定的监控内容输出文件名
  "monitor_out_flag": 1                 //
}
```

由壳程序外挂或间接启动的程序， 使用 monitor_standalone.exe启动

无依赖的程序 由monitor.exe启动

3. 对接方式

- 由壳程序启动的应用程序， 需自行将监控的内容， 按照格式输出到 自定义的文件中 （推荐每次重启清除该文件， 防止重复警告）
- 其他类型的应用程序， 可将监控的内容， 按照格式输出到标准输出（**推荐方式， 改动量小**）（c++: std::cout, c#: Console.writeLine, java: System.out.println）, 或按照格式输出到 自定义的文件中

格式定义为：

time@issue@description@severity@MONITORFLAG

时间@问题@描述@严重级别@MONITORFLAG

eg:

```
09/15/20 12:23:44@disconnected!@timeout after 15 seconds@FATAL@MONITORFLAG
09/15/20 13:25:04@disconnected!@timeout after 15 seconds@FATAL@MONITORFLAG
09/15/20 13:26:56@disconnected!@timeout after 15 seconds@FATAL@MONITORFLAG
09/15/20 13:30:06@disconnected!@timeout after 15 seconds@FATAL@MONITORFLAG
09/15/20 13:40:06@connected!@reconnect success@HIGH@MONITORFLAG
```

4. 健康状况检查

用于迅速判断一个应用程序是否健康 （应用逻辑层面）

如何对接?

4.1 开发人员 梳理应用内部逻辑， 定义程序健康状况的影响因素。

eg: 交易服务器的健康影响因素（factors）： f1[数据库连接状态] f2[与上手通信的连接状态] f3[前置的连接状态] fn[其他自定义的检查项] ……

定义程序整体健康状态的 规则， 一般情况为 健康状态 = f1 && f2 && f3 && …. && fn,

并在程序中实现该健康检查查询逻辑

4.2 对外提供 zabbix agent 查询的 http 接口， 接口定义如下

接口地址	**http://host/healthcheck**
请求方式	HTTP/POST
是否需要授权	否

请求示例

```
{
    "req_id": 1,    //请求号， 不校验， 用于respone带回
    "func_id": 10000,      //功能号， 标识此请求为检查整体健康状态
    “data”: {
        "AppName": "SGX_Communication"          //标识应用名称
    }
}
```

应答示例

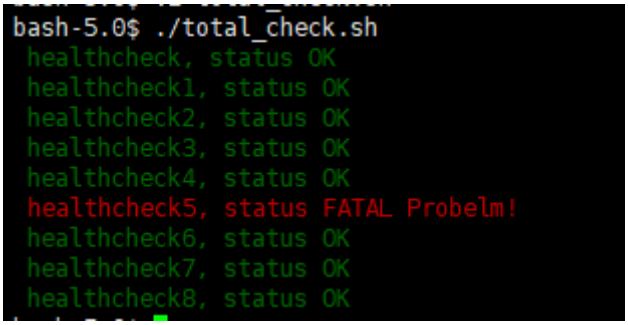
```
{
    "req_id": 1,    //请求号，使用 request中的值
    "func_id": 10000,      //功能号， 标识此应答为检查整体健康状态
    “data”: {
        "AppName": "SGX_Communication",          //标识应用名称
        "Health": 1          // 二值语义的健康状态     1: 健康    2: 异常
    }
}
```

4.3 全局应用健康状态的查询

- 应用程序提供http api **（开发人员提供）**
- host主机上 在agentd.conf中新增 自定义参数(userParam) 检查项， 并调用健康状况检查脚本 **（运维进行设置）**
- 重启 zabbix agent服务
- 使用**全局检查脚本** 查看 全局应用健康状态 **（已验证可用）**

zabbix agentd.conf 配置

全局检查脚本效果


bash-5.0$ ./total_check.sh
healthcheck, status OK
healthcheck1, status OK
healthcheck2, status OK
healthcheck3, status OK
healthcheck4, status OK
healthcheck5, status FATAL Probelm!
healthcheck6, status OK
healthcheck7, status OK
healthcheck8, status OK