统一数据接入 2.0API 说明-for C++

修订记录

修订日期	修订版本	修改描述
2011.05.09	1.0	正式稿第一版。
2011.06.03	1.1	修改了 baselog 的日志数据长度限制。
2011.06.09	1.2	修改了 CSEC 上报字段说明
2011.06.17	1.3	增加了 CSEC 留言发表类场景的上报示例
2011.07.11	1.4	为 CSEC 上报字段增加了 3 个 Webgame 扩展字段
2011.07.22	1.5	为上报字段“domain”增加了值“qqgame”
2011.08.01	1.6	多语言的数据上报字段说明相同，将上报字段说明独立出来，单独作为一份文档提供。
2011.09.21	1.7	接入互娱平台明星计划的应用初始化时 appname 必须传入"appoperlog"。
2011.10.08	1.8	对于互娱平台明星计划的应用， 数据上报函数（一次性）中参数 fallFlag 的值只能为" true "。
2011.10.13	1.9	对于互娱平台明星计划的应用， 上报经分，安全数据时 logtype 必须为“LT_BASE "。
2011.10.24	1.10	删除了关于多次上报一次提交的数据上报函数描述。
2011.12.01	1.11	删除了日志类型中的 LT_BASE = 2，将互娱明星计划应用的经分、安全数据上报与 LT_NORMAL= 0 合并； 增加了安全数据上报日志类型 LT_SECDATA = 4。
2011.12.19	1.12	原先只有接入互娱平台明星计划的应用初始化时

		appname 必须传入"appoperlog"，更改为所有应用初始化时，appname 都必须传入"appoperlog"。
2012.05.15	2.0	1. 修改了 AGENT 协议； 2. 兼容使用 1.0 API 的代码，只需替换库头文件重新编译即可； 3. 解决了 1.0 SDK 网络通讯的一些 BUG； 4. 对于实时转发的数据也写容灾（1.0 是不写容灾的），提升安全性； 5. 落地标识已经不在 API 控制，杜绝开发者写错的情况。
2012.10.18	2.1	LT_SECDATA = 4 对应的安全数据上报类型修改为反外挂数据上报类型。

修订流程

对于本文档中任何内容的增删改以及其它相关文档的创建，都应该知会系统开发人员，运维人员以及用户。

术语和缩略语清单：

缩略语	英文全名	中文解释
DC	DataCollector	统一数据接入

1. 统一数据接入2.0API总体说明

与统一数据接入 1.0 相比，统一数据接入 2.0 的优点为：

1. 修改 AGENT 协议，提高效率；
2. 兼容使用 1.0 API 的代码，只需替换库头文件重新编译即可；
3. 解决了 1.0 SDK 网络通讯的一些 BUG；

4. 对于实时转发的数据也写容灾（1.0 是不写容灾的），提升安全性；

5. 落地标识已经不在 API 控制，杜绝开发者写错的情况。

腾讯云平台为第三方应用开发者提供了通用日志数据上报接口，以及上报字段定义，称之为统一数据接入。

第三方开发者，以及腾讯云计算平台内部系统都通过调用统一的接口以及上报字段，进行日志数据上报：

- **第三方开发者上报应用的 CSEC 安全&实时数据分析字段**：使 CSEC 后台系统能够进行实时数据分析，为应用开发者制定反外挂打击策略等提供数据支持。
- **第三方开发者上报应用的 CBA 经营分析数据字段**：使 CBA 后台系统能够采集多维度的字段，为应用开发者展现更丰富的业务经营数据视图。
- **第三方开发者上报应用的模块间调用字段：** 使 CM 后台系统能够监控应用的各模块以及接口间调研那个情况，对每个请求的成功率、平均延迟指标的计算，及时发现调用失败的状况，帮助开发者分析和定位问题。
- **腾讯云计算平台的内部系统**也通过调用统一数据接入 API 接口，来实现基础数据，组件监控数据，自动化监控数据的采集，从而为开发者提供 CM 自动化监控服务，CBA 基础业务数据分析服务等云服务。

这样做大大简化了接口的调用门槛和上报流程，只保留了极少的需要开发者进行程序开发的操作，其他都由腾讯云计算平台自动采集完成，提升开发者接入云服务的效率。

2. 业务数据上报步骤

1. 定义日志数据上报对象。
2. 初始化日志数据上报对象。
3. 根据业务需求（例如是云安全接入，还是模调接入）选择需要上报的字段，构造日志数据。
    - 具体上报字段见：上报字段说明。
    - 采用 key-value 拼接的方式（例如 “key1=value1&key2=value2”）将上报的字段的 key-value 拼接起来。
    - 其中 value 中若含有”=”，”&”，”%” 时，需要先进行编码，可使用 API 中的字

符串编码函数。

4. 选择日志数据上报类型，见：日志类型定义。

5. 上报日志数据。

直接调用 write_baselog 函数。

注：兼容统一数据接入 1.0 版本的接口，如果使用过 1.0 版本的接口，请直接使用新版本的头文件和库重新编译即可，无须修改代码。

详情请见示例代码。

3. 上报字段说明

上报字段说明是通用的，不区分语言，因此独立提供一份文档。

请到统一数据接入页面下载最新的“统一数据上报字段说明”文档。

4. 日志类型定义

开发者在调用 write_baselog 函数上报数据时，需要传递“上报类型”参数。

上报类型定义如下：

```
typedef char logtype ;
const logtype LT_NORMAL    = 0;          // 经分数据上报
const logtype LT_MOD    = 1;          // 模调数据上报
const logtype LT_SECDATA    = 4;          // 反外挂数据上报
```

5. 库，头文件和命名空间

请到如下页面下载最新的 C++ SDK：

http://wiki.opensns.qq.com/wiki/%E7%BB%9F%E4%B8%80%E6%95%B0%E6%8D%AE%E6%8E%A5%E5%85%A5

头文件：dcapi_cpp.h

静态库：libdcapi_cpp.a

动态库：libdcapi_cpp.so

命名空间：namespace DataCollector;

6. 接口列表

6.1 定义日志上报对象

函数原型：

```
CLogger(char socketType = 0, bool needRsp = false);
```

接口说明：

定义日志上报对象，通常情况使用默认参数。

参数说明：

- socketType：

  0：使用unix socket链接dcagent。

  非0：使用tcp链接dcagent。

- needRsp：

  false：不需要从dcagent接收返回包。

  true：需要从dcagent接收返回包。

注：当短链接并发请求量很大时（超过3000次/秒），建议填true。

6.2 初始化业务数据上报对象

函数原型：

```
int init(string& logName, bool isModule = false);
```

接口说明：

用logName初始化日志上报对象。**只能调用一次。**

参数说明：

- logName： dc用来区分业务的标识。

注：第三方应用必须填”appoperlog "。

- isModule：是否为模调数据。

注：业务数据必须填false。

返回值说明：

- 0：成功。
- 非0：失败。详见错误码说明。

6.3 字符串编码函数

函数原型：

```
int encode(string& src_data, string& dst_data);
```

接口说明：

日志数据上报时采用key-value的方式，其中value中若含有”=”，”&”，”%” 时需要进行编码。本接口提供了简单的编码函数，对于字符”&”,”=”,”%”进行替换。

参数说明：

- src_data：原字符串。

- dst_data：输出参数，编码后的字符串。

返回值说明：

- 0：成功。
- 其它： 失败。详见错误码说明。

6.4 业务数据上报函数

函数原型：

```
int write_baselog(logtype type, string& data, bool fallFlag, unsigned int timestamp);
```

接口说明：

为了兼容1.0的旧代码，封装了数据上报函数。

参数说明：

- logtype：数据上报的类型，详见日志类型定义。
- data：已经编码后的数据，业务需要直接传递格式为 "k1=v1&k2=v2"的数据。
- fallFlag :参数无意义，用于兼容1.0版本。1.0版本中，该参数表示是否将数据落地。
- timestamp: 可以自定义时间戳(unix时间戳)，如果不填则使用系统当前时间。

返回值说明：

- 0：成功 。
- 其它：失败。详见错误码说明。

6.5 返回调用失败时的错误信息

函数原型：

```
string get_errmsg(void);
```

接口说明：

返回调用失败时的错误信息。

参数说明：

空

返回值说明：

调用失败时的错误信息。

6.6 业务数据上报接口（支持二进制数据上报）

函数原型：

```
int write_baselog(const char* data, unsigned int len, char logType, char proType, unsigned int timestamp)
```

接口说明：

上报业务数据，支持字符串与二进制协议。

参数说明：

- data：已经编码后的数据，业务需要直接传递格式为 "k1=v1&k2=v2"的数据（v1，v2已经做过编码）。

- len：数据的长度。
- logType：数据上报的类型，详见日志类型定义。
- proType： 0表示字符串；1表示二进制。
- timestamp: 可以自定义时间戳(unix时间戳)，如果不填则使用系统当前时间。

返回值说明：

- 0：成功 。
- 其它：失败。详见错误码说明。

7. 示例代码

请注意以下代码仅作示例，以便更清楚的演示日志数据上报的步骤。**不可直接copy编译**，

若有需要请参考SDK中的示例代码。

业务数据：

```
#include <stdio.h>
#include <stdlib.h>
#include "dcapi_cpp.h"
using namespace DataCollector;

int main(int argc ,char **argv)
{
        //Step1：定义新对象
        CLogger logger;
        string logname = "test";

        // Step2：用 logname 初始化对象
        logger.init(logname);
```

```
        int ret = 0;
        logtype type;

        // Step3 :构造 key-value 型数据，value 中不能含有“&”,“%”,"="特殊字符之一，否则要先进行编码
        //(1)根据上报字段说明，构造上报数据
        string data = "k1=1&k2=2&k3=kdjfdjkfdjkf&k5=888";
        //(2)value 中含有“&”,“%”,"="特殊字符之一的，要先进行编码
        string key = "k6";
        string value = "tt=tt";
        string resVaule;
        logger.encode(value, resVaule);
        //(3)将编码后的 key-value 追加到欲上报的数据之后
        data += "&";
        data += key;
        data += "=";
        data += resVaule;

        // Step4：选择上报的日志类型
        type = LT_NORMAL;

// Step5：上报日志数据
        ret = logger.write_baselog(type,data, true);
        if(ret!=0)
        {
            printf("Write_baselog data[%s] falied logtype[%d] ret[%d] err[%s]\n", data.c_str(), type,ret,logger.get_errmsg().c_str());
            return -1;
        }
        return 0;
}
```

8. 错误码说明

返回 0 表示 RET_SUCC，成功。

其他错误码含义见下表：

错误码	含义说明
1	RET_ERR_API_INIT 初始化错误。详见 FAQ 的说明。
2	RET_ERR_API_LOGLEN log 数据长度错误。详见 FAQ 的说明。
4	RET_ERR_API_SENDBUFFNULL 发送缓冲区错误。
5	RET_ERR_API_READBUFFNULL 接收缓冲区错误。
6	RET_ERR_API_SENDBUFFLEN 发送缓冲区长度错误。
7	RET_ERR_API_CONN unix socket 连接错误。详见 FAQ 的说明。
8	RET_ERR_API_SEND unix socket 发送错误。详见 FAQ 的说明。
9	RET_ERR_API_READ unix socket 接收错误。
10	RET_ERR_API_READLEN unix socket 接收包长错误。
10000	RET_ERR_SVR_SYSBUSY 统一数据接入 Agent 系统繁忙。
10001	RET_ERR_SVR_BUFFERLEN 数据长度超过 2k。详见 FAQ 的说明。
10002	RET_ERR_SVR_INVALIDAPPNAME 获取 Appname 失败。
10004	RET_ERR_SVR_WRITEDATA 写数据失败。
10005	RET_ERR_SVR_FETCHDATA 取数据失败。
10007	RET_ERR_SVR_READCONFIG 读配置出错。
10008	RET_ERR_SVR_LOGTYPE 无效的 logtype。
10009	RET_ERR_SVR_CMD

	无效的命令字。

9. FAQ

9.1 API容灾文件目录是什么？

/data/dcagent:　"/data/dcagent/msgdat.dat"

9.2 初始化API失败

问题描述：

调用 API 函数返回错误码 1。

根源分析：

多次调用 API 的初始化函数或者是在调用写日志相关函数之前没有调用过 API 初始化函数。

解决方案：

在调用写日志函数之前必先且只调用一次初始化函数。

9.3 日志数据长度有误

问题描述：

调用 API 写日志函数返回 2 或者 10001。

根源分析：

一次写入的数据长度超过 30KB（30720 字节）。

注：包括 API 自动添加的协议数据。

解决方案：

检查写入的数据长度。

9.4 API连接Agent失败

问题描述：

调用 API 写日志函数，返回错误码 7 或 8。

根源分析：

API 连接 Agent 失败，Agent 有可能没有启动，或者 unix socket 通讯存在问题。

解决方案：

检查系统中是否存在 dcagent 进程，确认 socket 通信地址（ /data/container2host/cdc）是否存在和有访问权限；

在 dcagent 的安装目录(一般为 /usr/local/services/CloudDCAgent_L5-1.0/)中的 env 目录，使用 check_env.sh(需要设置 gb 编码查看)检测当前 Agent 的运行环境有无问题；

如果个别机器的 dcagent 未启动，需要手动启动时，可执行 admin 目录下的./restart.sh all 来启动。