

统一数据接入 API 说明-for PHP

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"。
2012.10.18	1.13	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
CSEC	Cloud Security	腾讯云计算平台为开发者提供的安全服务
CBA	Cloud Business Analysis	腾讯云计算平台为开发者提供的经营数据分析服务
CM	Cloud Monitor	腾讯云计算平台为开发者提供的监控服务。

1. 统一数据接入API总体说明

[腾讯云计算平台](#)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 数据安全&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 业务数据字段**：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. 日志数据上报步骤

1. [构造](#)日志数据上报对象。
2. 根据业务需求（例如是云安全接入，还是模块接入）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](#)。
3. 选择日志数据上报类型，见：日志类型定义。
4. 上报日志数据。

直接调用 [write_base_log](#) 函数（Logtype 为 LT_NORMAL=0，即上报经分数据时，需要把 fallFlag 设置为 true，其余情况均设置为 false）。

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
```

注：

Logtype 为 LT_NORMAL= 0，即上报经分数据时，需要把 fallFlag 设置为 true，其余情况均设置为 false。

5. 库，头文件和命名空间

请到如下页面下载最新的 PHP 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>

PHP 扩展库：clogger.so

扩展库依赖的库：libdcapi_cpp.so

说明：

1. 上述.so 适用于 64 位系统。其中扩展 so 分两个版本，分别适用于 php5.3.* 和 php5.2.* 。

2. 调用统一数据接入的 PHP 接口前，需要为 php 添加扩展 clogger.so。首先修改配置文件 php.ini，添加扩展项。其次需将扩展文件放到 php.ini 所配置的扩展目录中。

修改 php.ini 的示例如下（假定扩展文件放置目录为 /usr/local/phpextension）：

```
extension_dir = "/usr/local/phpextension"

extension = clogger.so
```

修改方法：

- 修改 extension_dir 的值为“ /usr/local/phpextension”
- 添加一行 extension = clogger.so

注意：

为 php 添加扩展 clogger.so 后，php-fpm 或者 spawn-fcgi 等容器需要重启。

判断扩展加载成功的方法：

- 使用命令 php -m

在结果里面看到 clogger，表明 clogger.so 扩展已能正常加载。如下所示：

```
[PHP Modules]
bcmath
clogger
ctype
curl
date
```

- 使用命令 php --rc CLogger

在结果中能看到 CLogger 类的方法，即可确认扩展中接口已能被调用。如下所示：

```
Method [ <internal:clogger> public method encode ] {
}

Method [ <internal:clogger> public method write_base_log ] {
}
```

如果上面两步不成功，请确认是否按本节“说明”部分进行了操作。

6. 接口列表

6.1 构造日志上报对象

函数原型：

```
PHP_METHOD(CLogger, __construct)
```

接口说明：

用appname来构造日志上报对象。**只能调用一次。**

参数说明：

- appname：应用名，应用的唯一标识。

注：appname必须传入"appoperlog"。

返回值说明：

- 0：成功
- 1：失败。详见[错误码说明](#)。

6.2 字符串编码函数

函数原型：

```
PHP_METHOD(CLogger, encode)
```

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](#)。

6.3 数据上报函数

函数原型：

```
PHP_METHOD(CLogger, write_baselog)
```

接口说明：

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：是否将上报的日志数据落地。true: 落地，false: 不落地。logtype为

LT_NORMAL= 0，即上报经分数据时，需要把fallFlag设置为true，其余情况均设置为false。

返回值说明：

- 0：成功
- 其它：失败。详见[错误码说明](#)。

6.4 返回调用失败时的错误信息

函数原型：

```
PHP_METHOD(CLogger, get_errmsg)
```

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：

- 正常情况：调用失败时的错误信息字符串。
- 出错情况：返回FALSE。

7. 示例代码

请注意以下代码仅作参考，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。”

```
<?php

//Step1：用 appname 构造日志对象

//注： appname 必须传入"appoperlog"

$logger = new CLogger("huanshan");
```

```
date_default_timezone_set("prc");
```

```
$nowdate=date('Y-m-d H:i:s');
```

// Step2 : 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

```
$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

```
$key="k6";
```

```
$value = "tt=tt";
```

```
$resvalue="";
```

```
$res = $logger->encode($value,$resvalue);
```

```
echo "resvalue:". $resvalue. "\n";
```

//(3)将编码后的 key-value 追加到欲上报的数据之后

```
$data = $data."&". $key. "=" . $resvalue;
```

// Step3 : 选择上报的日志类型

```
$type = LT_NORMAL;
```

// Step4 : 上报日志数据

(注意 , 当 logtyp=LT_MOD , 即上报模调数据时 , failFlag 设置为 false , 即不需要数据落地)

```
$res = $logger->write_baselog($type,$data,TRUE);
```

```
echo "write_baselog res:". $res. "\n";
```

```
if($res != 0)
```

```
{
```

```
    $errmsg = $logger->get_errmsg();
```

```
    echo "errmsg:". $errmsg. "\n";
```

```
}
```

```
?>
```

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。详细说明见 FAQ。
2	RET_ERR_API_LOGLEN log 数据长度错误。详细说明见 FAQ。
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。详细说明见 FAQ。
8	RET_ERR_API_SEND unix socket 发送错误。详细说明见 FAQ。
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。详细说明见 FAQ。
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？

一次写入的数据 **Baselog**: "/data/dcagent/baselog.dat"

9.2 初始化错误

问题描述：

调用 API 函数返回错误码 1。

根源分析：

[声明日志上报对象](#)时，对象的初始化不成功。

解决方案：

重新声明一次对象。

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。