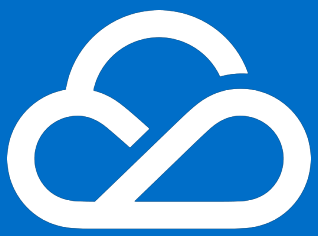
# 腾讯云游戏存储

# 开发者指南

# 产品文档



腾讯云

## 【版权声明】



## 【商标声明】



## 【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

文档目录

开发者指南

# 安装SDK

## 运行环境

Linux 2.6、Suse 32 位 / 64 位。

## Protobuf 安装

在安装和使用 Tcaplus Pb 前需安装 Protobuf，若已安装可忽略此节。
Protobuf 是 Google 推出的一种混合语言数据标准，是一种轻便的结构化数据存储格式。Tcaplus 系统支持使用 Protobuf 格式定义文件（.proto）定义数据表。在使用 Tcaplus Pb API 之前，需要在 GameSvr 服务器上安装 Protobuf。在此推荐使用源代码进行 Protobuf 安装，安装方法如下：

1. 准备 GameSvr 环境
   首先需要准备安装了 CentOS6- x86_64 或 CentOS7-x86_64 版本操作系统的 GameSvr。为了编译构建 Protobuf，需要安装以下软件，安装方法请自行查询：

   - autoconf
   - automake
   - libtool
   - curl (used to download gmock)
   - make
   - g++
   - unzip

2. 下载 Protobuf（syntax=” proto2” ）源码安装包
   单击下载源码安装包。

3. 解压源码安装包，并进入源码根目录下，以下为命令行操作。

   ```
   tar -xzvf protobuf-2.6.1.tar.gz
   cd ./protobuf-2.6.1
   ```

4. Configure 并指定安装路径前缀，推荐安装到路径 /usr/local/protobuf 下，以下为命令行操作。

```
./configure --prefix=/usr/local/protobuf
```

5. 编译并安装，以下为命令行操作。

```
make
make check
make install
```

6. 测试，使用 protoc 命令查看是否安装成功，以下为命令行操作。

```
protoc -version
```

如下图所示，安装成功。

```
[root@vm-10-125-32-21 /data/software_pkg/protobuf-2.6.1]# protoc --version
libprotoc 2.6.1
```

## SDK 下载

单击 SDK 下载。

## SDK 安装

tar –xzf<安装包路径> -C <安装目录>

## 校验

安装完成后，根目录结构如下表所示：

| 目录及文件 | 说明 |
| --- | --- |
| include/tcaplus_service/ | TCAPLUS 服务化 API 头文件 |
| lib/libtcaplusserviceapi.a | TCAPLUS 服务化 API 库文件 |
| include/tcaplus_service/protobuf/ | Protobuf API 头文件 |
| lib/libtcaplusprotobufapi.a | TCAPLUS Protobuf API 库文件 |
| examples/tcaplus/ProtoBuf | TCAPLUS Protobuf API 应用示例 |

## 运行 example 访问 Tcaplus

GameSvr 游戏服务器中对应数据访问逻辑的开发，可以参考 example 中的各接口实例。

1. 解压 Tcaplus Pb API 发布包

   tar -xzvf TcaplusPbApi3.18.0.152096.x86_64_release_20170712.tar.gz

2. 配置 Tcaplus 系统连接信息

   1. 在命令行输入

      cd TcaplusPbApi3.18.0.152096.x86_64_release_20170712/release/x86_64/examples/tcaplus/C++_common_for_pb2

      进入目录。
   2. 在命令行输入 vi common.h 修改 common.h 头文件，根据业务情况修改如下图片内容。
      DIR_URL_ARRAY：tcapdir 地址和端口号。
      DIR_URL_COUNT：tcapdir 个数。
      TABLE_NAME：需要访问的目标表。
      APP_ID：需要访问的 AppId。
      ZONE_ID：部署单元 ZoneId。
      SIGNATURE：业务 AppKey。

```
/*********************测试例子前需要用户手动修改的地方BEGIN*************************************
// 目标业务的tcapdir地址
static const char DIR_URL_ARRAY[][TCAPLUS_MAX_STRING_LENGTH] =
{
        "tcp://10.191.***.99:9999"
        "tcp://10.191.***.88:9999"
};

// 目标业务的tcapdir 地址个数
static const int32_t DIR_URL_COUNT = 1;
// 目标业务的表名
static const char * TABLE_NAME = "tb_online";
// 目标业务的App ID
static const int32_t APP_ID = 3;
// 目标业务的Zone ID
static const int32_t ZONE_ID = 1;
// 目标业务的业务密码
static const char * SIGNATURE = "*********";
/*********************测试例子前需要用户手动修改的地方END*************************************/
```

3. 修改环境配置设置文件

   在 TcaplusPbApi3.18.0.152096.x86_64_release_20170712/release/x86_64/examples/tcaplus 目录下有分别通过异步方式以及协程方式调用API的例子，现在以协程方式调用 Set 接口设置数据为例：

   命令行输入

   cd TcaplusPbApi3.18.0.152096.x86_64_release_20170712/release/x86_64/examples/tcaplus/C++_pb2_coroutine_simpletable/SingleOperation/set

   进入代码目录，协程方式 Set 例子的所有代码都在该目录中。修改 envcfg.env 文件，将 PROTOBUF_HOME 环境变量设置为本机 protobuf 的安装路径（--prefix指定），并将 TCAPLUS_HOME 环境变量设置为 Tcaplus Pb API 包下 release/x86_64 目录的绝对路径，如下图：

```
export PROTOBUF_HOME=/usr/loacl/protobuf;
export TCAPLUS_HOME=/my_some_dir/TcaplusPbApi3.18.0.123456.x86_64_release_20161124/release/x86_64/;
```

4. 设置环境变量

   在代码目录下执行

   source envcfg.env

   bash conv.sh

5. 编译二进制程序

   执行

make

命令编译 example 二进制,编译成功生成 mytest 可执行文件。

```
total 32
-rw-r--r-- 1 1002 users  416 Jul 12 16:40 conv.sh
-rw-r--r-- 1 1002 users  232 Aug 15 16:04 envcfg.env
-rw-r--r-- 1 1002 users 1766 Jul 12 16:40 main.cpp
-rw-r--r-- 1 1002 users  678 Jul 12 16:40 Makefile
drwxr-xr-x 2 root root  4096 Aug 15 16:05 mytest
-rw-r--r-- 1 1002 users 1047 Jul 12 16:40 readme.txt
-rw-r--r-- 1 1002 users  707 Jul 12 16:40 table_test.proto
-rw-r--r-- 1 1002 users  662 Jul 12 16:40 tlogconf.xml
```

6. 执行二进制程序

   命令行输入

   ./mytest

   ,执行二进制程序。执行结果将在命令行标准输出中显示，若遇到错误，请查看代码目录下的 tcaplus_pb.log 日志文件。

# API手册

## API 简介

Tcaplus 化 API 是应用访问游戏存储服务集群的入口，是应用存取游戏存储服务集群中业务数据的编程接口。

## API 流程

应用在使用 API 时，需要提供 AppId，认证密码，一组入口 URL，以及已经创建好的表名称和 ZoneId 等信息。其中，前三项是应用申请接入游戏存储服务成功后得到的，后两项是业务在游戏存储管理页面上创建表结构时，提供和确定下来的。使用游戏存储服务化 API，应用可以同时操作属于 AppId 的多个数据表。

## API 模块

用户可以定义 Protobuf 表，把表的协议传到 Tcaplus 系统中进行加表，即可通过 Tcaplus Protobuf API 进行表数据的读写操作，当前 Protobuf API 支持的操作如下表：

| 操作 | 功能描述 |
| --- | --- |
| GET | 根据用户传的单个的 Key 获得单条 Value 信息 |
| BATCHGET | 根据用户传的多个的Key 获得多条 Value 信息 |
| SET | Key 对应记录存在则更新数据，否则插入数据请求 |
| DEL | 根据 Key 删除对应的记录 |
| INC | 指定字段的值（整数型）增加指定值 |
| FIELDSET | 更新指定字段(单个或多个)的值 |
| FIELDGET | 获取指定字段(单个或多个)的值 |

## Protobuf API 约定

Tcaplus 需要定义表的 primarykey 等信息，扩展 tcaplusservice.optionv1.proto 存在于 Tcaplus 系统中，用户只需引用，建表时不用上传，具体内容如下：

```
extend google.protobuf.MessageOptions
{
  optional string tcaplus_primary_key = 60000; //Tcaplus Primary Key
```

```
  repeated string tcaplus_index = 60001; //Tcaplus Index
}
extend google.protobuf.FieldOptions
{
  optional uint32 tcaplus_size = 60000; // Tcaplus field size
  optional string tcaplus_desc = 60001; // Tcaplus description
}
```

完整文件可在目录 release\x86_64\include\tcaplus_service\tcaplusservice.optionv1.proto 中找到。

1. 表名要以字母或下载线开头，不能超过 31 个字段，不能有除数字，字母，下划线之外的特殊字段。
2. 各字段的名称要以字母或下划线，protobuf 已经限制。
3. 主键最多 4 个字段，必须是 required 类型，打包后长度不能超过 1022 字节。
4. value 打包后不能超过 256 KB, 同时整个记录打包不能超过 256 KB。
5. 除了 key 字段，至少有一个 value 字段。value 字段上限以 protobuf 为准。
6. 表默认是 generic 类型。但对外不显示 generic 类型。
7. key 字段当前只能是 protobuf 规定的标量类型（Scalar Value Type）,不能包括其它复合类型，自定义类型等。

## API 常用接口

```
int Get(::google::protobuf::Message &msg);

  @brief 根据用户输入的 msgs 中的 key 值，批量获取 msg 消息的字段值，并填充到 msgs 中.
  @param [INOUT] msgs 用户输入的 key 值列表，返回指定字段填到 msgs 中
  @retval <0 失败，返回对应的错误码。
  @retval 0 成功。至少有一个字段查询成功才会返回 0。

int BatchGet(std::vector< ::google::protobuf::Message * > &msgs);

  @brief 根据用户输入的 msg 中的 key 值，删除 msg。
  @param [IN] msg 用户输入的 key 值，返回指定字段填到 msg 中。
```

@retval <0 失败，返回对应的错误码。
@retval 0 成功。至少有一个字段查询成功才会返回 0。

int Del(const ::google::protobuf::Message &msg);

@brief 根据用户输入的 msg 中的 key 值，和 dottedpaths 指定的字段名称，获取指定字段的值，并填充到 msg 中。
@param [INOUT] msg 用户输入的 key 值，返回指定字段填到 msg 中。
@param [IN] dottedpaths 字段名称的点分嵌套字符串集。
@param [OUT] failedpaths 返回查找失败的字段名称的点分嵌套字符串集。
@retval <0 失败，返回对应的错误码。
@retval 0 成功。至少有一个字段查询成功才会返回 0。

int Set(const ::google::protobuf::Message &msg);

@brief 根据用户输入的 msg 中的 key 值和 values 增量值，和 dottedpaths 指定的字段名称，增加指定字段的值。字段为数值型变量。
@param [INOUT] msg 用户输入的 key 值，返回增量字段的结果值更新到 msg 中。
@param [IN] dottedpaths 字段名称的点分嵌套字符串集。
@retval <0 失败，返回对应的错误码。表示没有任何字段更新。
@retval 0 成功。全部字段更新成功。

int FieldInc(::google::protobuf::Message &msg, const std::set &dottedpaths);

@brief 根据用户输入的 msg 中的 key 值，和 dottedpaths 指定的字段名称，更新指定字段的值。服务端不存在的值会追加进去。
@param [IN] msg 用户输入的 key 值，返回指定字段填到 msg 中。
@param [IN] dottedpaths 字段名称的点分嵌套字符串集。
@retval <0 失败，返回对应的错误码。表示没有任何字段更新。
@retval 0 成功。全部字段更新成功。

int FieldSet(::google::protobuf::Message &msg, const std::set &dottedpaths);

@brief 根据用户输入的 msg 中的 key 值，和 dottedpaths 指定的字段名称，获取指定字段的值，并填充到 msg 中。
@param [INOUT] msg 用户输入的 key 值，返回指定字段填到 msg 中。
@param [IN] dottedpaths 字段名称的点分嵌套字符串集。
@param [OUT] failedpaths 返回查找失败的字段名称的点分嵌套字符串集。
@retval <0 失败，返回对应的错误码。
@retval 0 成功。至少有一个字段查询成功才会返回 0。

## 规则与约束

Get 操作约束：
1. 不能查询特定版本号的记录。
2. 如果待查询的字段原记录中不存在，则会返回字段默认值。

BatchGet 操作约束:
1. 批量查询操作必须经由 tcaproxy 进行消息路由处理，tcapsvr 进程不支持批量查询操作。
2. 一个批量查询请求返回一个批量查询结果，批量查询的超时时间为 10 秒。
3. 批量查询结果记录数同请求中记录数，查询不存在或者查询失败的记录也会在结果集中存在一条空的记录，因此需要通过 FetchRecord 进行记录获取。
4. 批量查询结果集总大小不能超过 256 K，否则超过大小记录内容会无法返回，会返回空记录内容。
5. 批量查询结果集返回的顺序与请求的记录顺序不保证一致。