行业事业部

长城IM通信平台协议设计

V1.0

修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **修订人** | **修订说明** |
| 2014-07-29 | V1.0 | 白叶 | 初稿 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

[修订记录 2](#_Toc394675148)

[目录 3](#_Toc394675149)

[1. 概述 4](#_Toc394675150)

[1.1 背景 4](#_Toc394675151)

[1.2 Protocol Buffers简介 5](#_Toc394675152)

[1.3 实现方式 6](#_Toc394675153)

[1.4 通信链路 6](#_Toc394675154)

[2. 命令及协议 7](#_Toc394675155)

[2.1 PROTO定义 7](#_Toc394675156)

[2.1.1 GwChat.Base 7](#_Toc394675157)

[2.1.2 GwChat.User 8](#_Toc394675158)

[2.1.3 GwChat.Cmd 9](#_Toc394675159)

[2.1.4 GwChat.Template 11](#_Toc394675160)

[2.2 请求和响应包结构 13](#_Toc394675161)

[2.2.1 请求包结构 13](#_Toc394675162)

[2.2.3 响应包结构 13](#_Toc394675163)

[2.3 请求和响应命令 14](#_Toc394675164)

[2.3.1 心跳 14](#_Toc394675165)

[2.3.2 Push 14](#_Toc394675166)

[2.3.3 Push确认 14](#_Toc394675167)

[2.3.4 发送消息 14](#_Toc394675168)

[2.3.5 收取消息 15](#_Toc394675169)

[2.3.6 确认收取消息 15](#_Toc394675170)

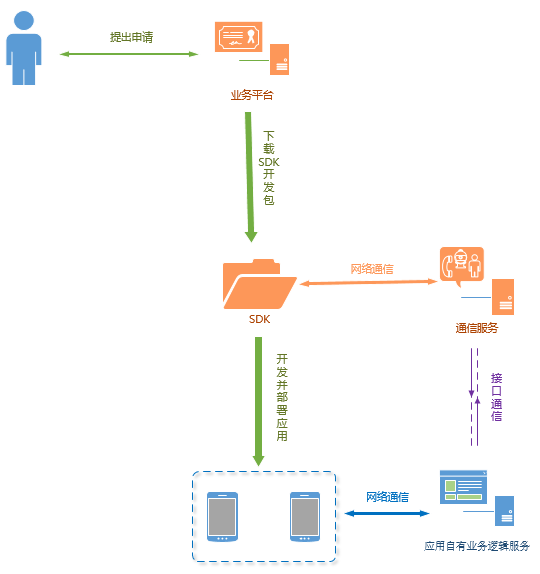
[2.3.7 命令时序示意 16](#_Toc394675171)

# 概述

## 1.1 背景

即时通信（IM）功能已逐渐成为移动应用不可或缺的功能之一，为方便公司的移动应用都能方便、快速的集成IM功能，一个轻量级的、通用的即时通信平台应运而生。

本平台主要由客户端SDK、通讯服务、业务平台三部分组成，结构及流程示意图：



本文档主要描述 **SDK** 和 **通信服务** 之间的通信协议设计

## 1.2 Protocol Buffers简介

Protocol Buffers是一种轻便高效的结构化数据存储格式，可以用于结构化数据串行化，即序列化。很适合做数据存储或 RPC 数据交换的一种格式。可用于通讯协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据的一种格式。可以使用它来持久化数据或者序列化成网络传输的数据。相比较JSON & XML技术而言，该技术的一个明显特点就是更加节省空间（以二进制流存储）、速度更快以及更加灵活。是Google 公司内部的混合语言数据标准。参见：

<http://code.google.com/p/protobuf/>

<http://www.ibm.com/developerworks/cn/linux/l-cn-gpb/index.html>

**Protocol Buffers基础数据类型对照表：**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| .proto | C++ | C# | Java | Notes |
| double | double | double | double |  |
| float | float | float | float |  |
| int32 | int32 | int | int | Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint32 instead. |
| int64 | int64 | long | long | Uses variable-length encoding. Inefficient for encoding negative numbers – if your field is likely to have negative values, use sint64 instead. |
| uint32 | uint32 | uint | int | Uses variable-length encoding. |
| uint64 | uint64 | ulong | long | Uses variable-length encoding. |
| sint32 | int32 | int | int | Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int32s. |
| sint64 | int64 | long | long | Uses variable-length encoding. Signed int value. These more efficiently encode negative numbers than regular int64s. |
| fixed32 | uint32 | int | int | Always four bytes. More efficient than uint32 if values are often greater than 228. |
| fixed64 | uint64 | long | long | Always eight bytes. More efficient than uint64 if values are often greater than 256. |
| sfixed32 | int32 | int | int | Always four bytes. |
| sfixed64 | int64 | long | long | Always eight bytes. |
| bool | bool | bool | boolean |  |
| string | string | string | String | A string must always contain UTF-8 encoded or 7-bit ASCII text. |
| bytes | string | byte[] | ByteString | May contain any arbitrary sequence of bytes. |

## 1.3 实现方式

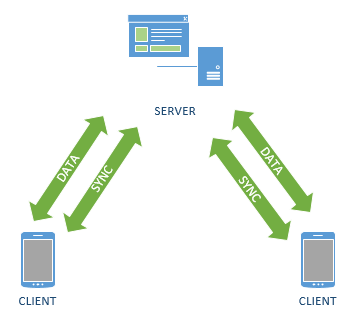
本协议采用网络数据流的应答方式，若基于Socket则采用标准二进制数据通讯，~~若基于HTTP则采用 POST方式通讯~~，均以Protocol Buffers数据格式来进行交互响应。

## 1.4 通信链路

分 同步链路(SYNC) 和 数据链路(DATA)

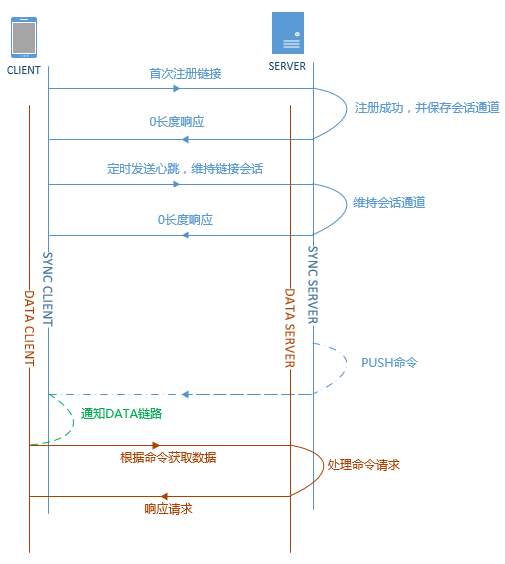
Sync用于维持客户端和服务器端的通讯会话，由客户端来实现网络纠错及失败重连的过程

Data链路即标准的Client🡪Server访问模式，实现定向按需连接



**通信流程：**

1. 客户端判断当前网络类型，通过Sync链路到服务器端注册，打通Server Push通道（长连接）；
2. 客户端操作若需获取相应数据则通过Data链路来传输（短连接）；
3. 服务器端需要PSUH消息或数据时通过Sync链路来传递相关命令，客户端获取命令后再通过Data链路来进行通讯；



# 命令及协议

## 2.1 PROTO定义

### 2.1.1 GwChat.Base

**文件：** Base.proto

**内容：**

option optimize\_for = LITE\_RUNTIME;

option java\_package = "GwChat";

option java\_outer\_classname = "Base";

//命名空间

package GwChat.Base;

//请求基类数据结构描述

message RequestBase

{

optional string AppKey = 1; //应用认证KEY，用于标示当前应用

optional string Version = 2; //当前SDK版本，用于区分不同版本的协议差异

optional string Command = 3; //请求命令

optional bytes Data = 4; //与请求命令对应的结构数据经PB序列化后的内容

optional bytes Attach = 5; //请求附加内容

}

//响应基类数据结构描述

message RespondBase

{

optional Status Result = 1[default=Succeed]; //响应状态

optional string Code = 2; //响应状态码

optional string Message = 3; //响应描述

optional string Command = 4; //响应命令

optional bytes Data = 5; //与响应命令对应的PB序列化数据

optional bytes Attach = 6; //响应附加内容

}

//请求响应状态枚举

enum Status

{

Failed = 0; //失败

Succeed = 1; //成功

}

### 2.1.2 GwChat.User

**文件：** User.proto

**内容：**

option optimize\_for = LITE\_RUNTIME;

option java\_package = "GwChat";

option java\_outer\_classname = "User";

//命名空间

package GwChat.User;

//用户信息数据结构描述

message Member

{

optional MemberType Type = 1[default = Normal]; //成员类型

optional uint32 Id = 2; //成员ID，唯一值。

optional string Name = 3; //成员用户名，唯一值。

optional string NickName = 4; //成员昵称

optional string HeadIcon = 5; //用户头像（群图标）HTTP路径

}

//用户类型枚举

enum MemberType

{

Normal = 0; //普通用户

System = 1; //普通系统用户（和正常用户一般可接收、发送消息）

Push = 2; //系统消息推送用户（只单向推送消息至客户端）

Group = 4; //普通群组

}

### 2.1.3 GwChat.Cmd

**文件：** Cmd.proto

**内容：**

option optimize\_for = LITE\_RUNTIME;

option java\_package = "GwChat";

option java\_outer\_classname = "Cmd";

//命名空间

package GwChat.Cmd;

import "User.proto";

//消息数据结构描述

message Message

{

optional string Id = 1; //消息Id，服务器分配，全局唯一

optional GwChat.User.Member Sender = 2; //消息发送人

optional GwChat.User.Member Receiver = 3; //消息接收人

optional string Time = 4; //消息发送时间

optional string TypeName = 5; //消息类型名称

optional bytes Content = 6; //消息内容数据

}

//收取消息命令和消息列表通用信息数据结构描述

message Messages

{

optional GwChat.User.Member Owner = 1; //当前收取消息的所有者

optional bytes Range = 2; //收取消息范围

repeated UserMessages MessageList = 3; //以发送人为单位的用户消息列表

}

//以发送人为单位的消息列表数据结构描述

message UserMessages

{

optional GwChat.User.Member User = 1; //当前消息列表的发送人

repeated Message MessageList = 2; //消息列表

}

//标记为已收消息命令数据结构描述

message Received

{

optional GwChat.User.Member Owner = 1; //命令发起人

optional bytes Range = 2; //标记的消息范围

}

//PUSH内容类数据结构描述

message Push

{

optional string Id = 1; //PUSH序号，用于确认回传

optional GwChat.User.Member Receiver = 2; //PUSH命令接收对象

optional string Command = 3; //PUSH命令

optional bytes Data = 4; //与请求命令对应的数据(PB序列化)

}

//心跳命令数据结构描述

message HeartBeat

{

optional uint32 Id = 1; //本次会话中心跳序号从0开始每次递增1

optional GwChat.User.Member Sender = 2; //命令发起人

optional string DeviceToken = 3; //设备标识

}

### 2.1.4 GwChat.Template

**文件：** Template.proto

**内容：**

option optimize\_for = LITE\_RUNTIME;

option java\_package = "GwChat";

option java\_outer\_classname = "Template";

//命名空间

package GwChat.Template;

//文本消息（含表情）模板数据结构描述

message TextMessage

{

optional string Text = 1; //消息内容

}

//图片消息模板数据结构描述

message ImageMessage

{

optional bytes Bin = 1; //图片数据

optional string Name = 2; //图片文件名

optional string Url = 3; //大图完整HTTP URL路径

}

//声音消息模板数据结构描述

message VoiceMessage

{

optional bytes Bin = 1; //声音数据

optional uint32 Duration = 2; //声音时长，单位：秒

}

//位置消息模板数据结构描述

message LocationMessage

{

optional string Address = 1; //地址

optional double Longitude = 2; //经度

optional double Latitudes = 3; //纬度

}

//新闻消息模板数据结构描述

message NewsMessage

{

repeated News NewsList = 1; //新闻列表

//新闻消息中单条新闻内容数据结构描述

message News

{

optional string Title = 1; //新闻标题

optional string Intro = 2; //新闻内容简介

optional string Time = 3; //新闻发布时间

optional string ImgUrl = 4; //新闻图片HTTP路径

optional string NewsUrl = 5; //新闻详情内容WEB页面HTTP路径

}

}

//调用设备接口消息模板数据结构描述

message APIMessage

{

optional string Source = 1; //原始内容

repeated APIKey KeyList = 2; //可执行API的关键内容列表

//可执行API的关键内容数据结构描述

message APIKey

{

optional string Key = 1; //关键字

optional string RealKey = 2; //传递给API的实际内容

optional APIType Action = 3[default=Dialing]; //API类型

optional string Subject = 4; //主题参数

optional string Body = 5; //内容参数

}

//可调用的系统API类型枚举

enum APIType

{

Dialing = 0; //拨打电话

Sms = 1; //发送短信

Email = 2; //发送电子邮件

Browser = 3; //浏览器（外部）

WebView = 4; //浏览器控件

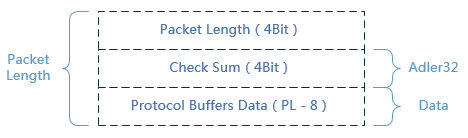
HiddenMessage = 5; //发送无记录消息

}

}

## 2.2 请求和响应包结构

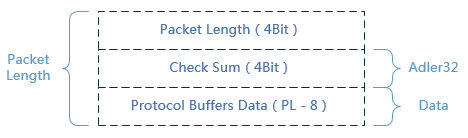
### 2.2.1 请求包结构



PL = Packet Length

Protocol Buffers Data = ProtoBufSerialize<RequestBase>()

### 2.2.3 响应包结构



PL = Packet Length

Protocol Buffers Data = ProtoBufSerialize<RespondBase>()

## 2.3 请求和响应命令

### 2.3.1 心跳

**说明：**打通和维持SYNC会话通道

**[上行]**

**命令：**GwChat.Cmd.HeartBeat

**参数：**GwChat.Cmd.HeartBeat.Id 从0开始，每次递增1

**[下行]**

响应命令和数据为空

**注：**收到响应后，心跳计时器归0，重新开始计时

### 2.3.2 Push

**说明：**服务器通过SYNC主动下发命令至客户端

**[下行]**

**命令：**GwChat.Cmd.Push

**注：**每收到一次PUSH，将心跳计时器归0，重新开始计时

### 2.3.3 Push确认

**说明：**通过DATA链路回复服务器，确认收到该Push

**[上行]**

**命令：**GwChat.Cmd.Push

**参数：**Id、Receiver 直接复制下行包字段值，Command、Data置空

**[下行]**

响应命令和数据为空

### 2.3.4 发送消息

**说明：**发送消息

**[上行]**

**命令：**GwChat.Cmd.Message 除Id外，其余字段应该全部正确赋值

**[下行]**

**命令：**GwChat.Cmd.Message 除Id 外，其余字段全部置空

### 2.3.5 收取消息

**说明：**收取消息

**[上行]**

**命令：**GwChat.Cmd.Messages

**[下行]**

**命令：**GwChat.Cmd.Messages

### 2.3.6 确认收取消息

**说明：**确认本次收取消息成功，服务器重置消息状态，确保不会丢失和重复发送

**[上行]**

**命令：**GwChat.Cmd.Received

**[下行]**

响应命令和数据为空

### 2.3.7 命令时序示意

