**设备终端对接说明**

**文档修订记录**

| **版本** | **变化**  **状态** | **简要说明（变更内容和变更范围）** | **变更日期** | **变更人** | **评审日期** | **评审人** |
| --- | --- | --- | --- | --- | --- | --- |
| v1.0 | 新建 | 设备终端对接说明 | 2017/03/20 | 刘成 |  |  |
| V1.1 | 新增 | 新增控制指令CLOSE,  关闭播放器 | 2017/03/21 | 刘成 |  |  |
| V1.2 | 新增 | 新增播放进度事件，新增下载完成确认事件,参数修改 | 2017/04/7 | 刘成 |  |  |
| V1.3 | 新增 | 新增播放进度通知属性PlayStatus | 2017/04/19 | 刘成 |  |  |
| V1.4 | 修改 | 修改下载通知结构，新增下载通知消息结构体MessageInfo | 2017/06/13 | 刘成 |  |  |
| V1.5 | 修改 | 注册事件新增属性设备名称equipmentName，  新增设置设备名称事件 | 2017/07/12 | 刘成 |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

※ 变化状态：新建、修改、增加、删除

**目 录**

[1 概述 1](#_Toc29808)

[2 protobuf文件说明 1](#_Toc32569)

[2.1 事件传递类型枚举 1](#_Toc14481)

[2.2 控制指令枚举 1](#_Toc30310)

[2.3 响应状态枚举 2](#_Toc23166)

[2.4 播放状态枚举 2](#_Toc23166)

[2.5 下周通知内容结构体 2](#_Toc23166)

[3 终端与控制端交互 2](#_Toc987)

[3.1 请求参数说明 2](#_Toc15238)

[3.2 响应参数说明 2](#_Toc6964)

[3.3 请求响应事件 2](#_Toc16967)

[3.3.1](#_Toc11302) **[心跳事件](#_Toc11302)** [2](#_Toc11302)

[事件类型：HEARTBEAT 2](#_Toc31987)

**[3.3.2 设备注册事件](#_Toc7229)** [2](#_Toc7229)

[3.3.3 设置设备名称事件 3](#_Toc4733)

[3.3.4 播放进度事件 3](#_Toc22484)

[3.3.5 下载完成确认事件 3](#_Toc25152)

[3.4 通知事件 3](#_Toc31640)

[3.4.1 控制指令](#_Toc28138)**[通知事件](#_Toc28138)** [3](#_Toc28138)

[3.4.2 下载](#_Toc25426)**[通知事件](#_Toc25426)** [3](#_Toc25426)

[3.4.3](#_Toc18351) **[删除通知事件](#_Toc18351)** [3](#_Toc18351)

[3.4.4 播放进度](#_Toc4045)**[通知事件](#_Toc4045)** [3](#_Toc4045)

[4 接口 4](#_Toc20231)

[4.1 心跳事件 4](#_Toc9484)

[4.1.1 接口名称 4](#_Toc6289)

[4.1.2 调用说明 4](#_Toc2872)

[4.1.3 返回说明 4](#_Toc459)

[4.2 注册事件 4](#_Toc20458)

[4.2.1 接口说明 4](#_Toc14266)

**[4.1.2 调用说明](#_Toc4787)** [4](#_Toc4787)

[4.1.3 返回说明 5](#_Toc28609)

[4.3 设备重连事件 5](#_Toc24346)

[4.3.1 接口说明 5](#_Toc19506)

[4.3.2 调用说明 5](#_Toc10095)

[4.3.3 返回说明 5](#_Toc29277)

[4.4 播放进度事件 6](#_Toc28433)

[4.4.1 接口说明 6](#_Toc17965)

[4.4.2 调用说明 6](#_Toc26659)

[4.4.3 返回说明 6](#_Toc8080)

[4.5 下载完成确认 7](#_Toc10188)

[4.5.1 接口说明 7](#_Toc12349)

[4.5.2 调用说明 7](#_Toc28009)

[4.5.3 返回说明 7](#_Toc17562)

[5 通知 8](#_Toc7260)

[5.1 控制指令通知事件 8](#_Toc8391)

[5.1.1 通知说明 8](#_Toc17764)

[5.2 下载通知事件 8](#_Toc800)

[5.2.1 通知说明 8](#_Toc17313)

[5.3 删除通知事件 8](#_Toc24500)

[5.3.1 通知说明 8](#_Toc21159)

[5.4 播放进度通知事件 9](#_Toc17534)

[5.4.1 通知说明 9](#_Toc28246)

[6 示例： 10](#_Toc5224)

[6.1 注册请求示例 10](#_Toc4690)

[eventId：事件ID 10](#_Toc30558)

[Types.REGISTER](#_Toc24141)*[： 请求事件类型](#_Toc24141)* [10](#_Toc24141)

[6.2 消息通知示例 11](#_Toc4088)

**[6.2.1 控制指令通知结构体](#_Toc3454)** [11](#_Toc3454)

概述

控制端软件连接终端采用netty+protobuf技术开发，所以本文档将主要说明控制端与终端端交互数据的protobuf格式文档内容。

Netty：JBOSS提供的一个java开源框架，提供异步的、基于事件驱动的网络应用框架，用以快速开发高性能、高可靠的网络服务器和客户端程序。

Protobuf：全称 protocol buffer，是google提供的一种数据交换格式，它独立于语言，独立于平台。Google提供了多种语言的实现，如java、c#、go、c++及python等。由于它是一种二进制格式，比使用xml进行数据交换快很多。作为一种效率和兼容性都很优秀的二进制数据传输格式，可以用于诸如网络传输、配置文件、数据存储等诸多领域。

附件：MessageProto.proto 为protobuf定义的消息协议



protobuf文件说明

## 事件传递类型枚举

enum Types {

HEARTBEAT =0; //心跳

REGISTER =1; //设备注册

CTRL\_DICTATE =2;//控制指令

DOWNLOAD=3; //下载

DELETE=4; //删除

RECONNECT=5; //重连

PLAY\_PROGRESS=6; //播放

CTRL\_DICTATE\_NOTICE=7; //控制指令通知

DOWNLOAD\_NOTICE =8; //下载通知

DELETE\_NOTICE=9; //删除通知

DOWNLOAD\_ACK=10; //下载确认

PLAY\_PROGRESS\_NOTICE=11; //播放通知

GET\_PLAY\_PROGRESS=12; //获取播放进度

SET\_PLAY\_PROGRESS=13; //设置播放进度

## 控制指令枚举

enum Dictate{

START=0;//开始

PLAY=1; //播放

STOP=2; //暂停

END=3; //结束

CLOSE=4; //关闭播放器

}

## 响应状态枚举

enum RespStatus{

SUCCESS //成功

FAILURE //失败

}

## 播放状态枚举

enum PlayStatus {

PLAYI\_ING //播放

PLAY\_STOP //暂停

PLAY\_END //结束

}

## 下载通知内容结构体

Message MessageInfo{

required string fileId=1; //文件ID

required string fileSize=2; //文件大小

optional string fileName=3; //文件名称

optional string title =4; //消息标题

optional string content =5; //消息内容

optional string imageUrl=6; //图片

}

# 终端与控制端交互

## 请求参数说明

Types type --请求的事件类型（必填）

## 响应参数说明

Types type --返回用户操作对应的请求类型（必填）

RespStatus status --返回成功失败状态（必填）

string statusCode --返回相应状态码

string statusDesc --返回响应提示信息

## 请求响应事件

**request请求：终端发送请求至控制端**

**Response响应：控制端根据终端发送的事件请求返回的响应**

### **心跳事件**

事件类型：HEARTBEAT

request请求：heartBeatRequest(终端发送至控制端的心跳请求)

Response响应: 无

### **设备注册事件**

事件类型：REGISTER

request请求：registerRequest (设备注册请求)

Response响应:registerResponse(设备注册响应)

### 设置设备名称事件

事件类型：SET\_EQUIPMETN\_NAME

request请求：SaveEquipmentNameRequest (设置设备名称事件请求)

Response响应：SaveEquipmentNameResponse (设置设备名称事件响应)

### 播放进度事件

事件类型：PLAY\_PROGRESS

request请求：PlayProgressRequest(播放进度请求)

Response响应：PlayProgressResponse(播放进度响应)

### 下载完成确认事件

事件类型：DOWNLOAD\_ACK

ACK请求：DownloadAck(下载完成确认)

## 通知事件

### 控制指令**通知事件**

通知事件类型：CTRL\_DICTATE\_NOTICE

通知事件：CtrlDictateNotice (控制端指令通知)

### 下载**通知事件**

通知事件类型：DOWNLOAD\_NOTICE

通知事件：DownLoadNotice (下载通知)

### **删除通知事件**

通知事件类型：DELETE\_NOTICE

通知事件：DeleteNotice (发送消息通知)

### 播放进度**通知事件**

通知事件类型：PLAY\_PROGRESS\_NOTICE

通知事件：PlayProgressNotice (播放通知)

# 接口

## 心跳事件

### 接口名称

Protobuf协议请求接口：

HeartBeatRequest

**请求方式：**

IP+端口

#### **请求事件类型：** **HEARTBEAT**

##### 参数：

无;

接口的描述：发送心跳包 设备终端每隔X秒向服务端发送心跳指令，控制中心收到心跳指令认为设备在线。 一段时间内 未收到设备心跳指令则认为设备停止使用

### 调用说明

参考示例： [6.1注册请求示例](#_发送请求示例)

### 返回说明

返回示例：

服务端无返回

## 注册事件

### 接口说明

Protobuf协议请求接口：

RegisterRequest

**请求方式：**

IP+端口

##### 请求事件类型： Register

##### 参数：

eventId：终端传至服务端的事件id

equipmentId：终端设备唯一标示

equipmentName ：设备名称

**接口的描述：**

终端设备注册至控制中心，表示终端在线

**4.1.2 调用说明**

参考示例[5.1 注册请求示例](#_发送请求示例)

### 4.1.3 返回说明

返回示例：

type: REGISTER

status: SUCCESS

registerResponse {

eventId: "1"

equipmentId: "88d77aa0-a116-4715-ae78-00ff8ea4c5da"

}

返回参数说明：

|  |  |  |
| --- | --- | --- |
| 返回参数 | 返回参数类型 | 返回参数意义 |
| equipmentId | String | 注册至控制中心的设备唯一标示 |
| eventId | string | 事件id |

## 设置设备名称事件

### 接口说明

Protobuf协议请求接口：

SaveEquipmentNameRequest

**请求方式：**

IP+端口

##### 请求事件类型： SET\_EQUIPMETN\_NAME

##### 参数：

eventId：终端传至服务端的事件id

equipmentId：终端设备唯一标示

equipmentName 设备名称

**接口的描述**：

设备终端需要检测到服务器连接中断时。可发起重连事件

### 调用说明

参考示例[6.1 注册请求示例](#_发送请求示例)

### 返回说明

返回示例：

type: SET\_EQUIPMETN\_NAME

status: SUCCESS

SaveEquipmentNameResponse{

eventId: "1"

equipmentId: "aaaaaaaaaaaaaa"

equipmentName："设备名称"

}

返回参数说明：

|  |  |  |
| --- | --- | --- |
| 返回参数 | 返回参数类型 | 返回参数意义 |
| eventId | string | 事件id |
| equipmentId | String | 注册至控制中心的设备唯一标示 |
| equipmentName | String | 设备名称 |

## 播放进度事件

### 接口说明

Protobuf协议请求接口：

PlayProgressRequest

**请求方式：**

IP+端口

##### 请求事件类型： PLAY\_PROGRESS

##### 参数：

eventId：终端传至服务端的事件id

equipmentId：终端设备唯一标示

**接口的描述**：

设备终端连接上服务端后 客户端可以通过此接口获取播放进度

### 调用说明

参考示例[6.1 注册请求示例](#_发送请求示例)

### 返回说明

返回示例：

type: PLAY\_PROGRESS

status: FAILURE

playProgressResponse {

eventId: "1"

fileId: ""

position: ""

}

返回参数说明：

|  |  |  |
| --- | --- | --- |
| 返回参数 | 返回参数类型 | 返回参数意义 |
| eventId | string | 事件id |
| fileId | string | 文件唯一标示 |
| position | string | 文件播放位置 |

## 下载完成确认

### 接口说明

Protobuf协议请求接口：

DownloadAck

**请求方式：**

IP+端口

##### 请求事件类型： DOWNLOAD\_ACK

##### 参数：

eventId：终端传至服务端的事件id

equipmentId：终端设备唯一标示

fileIds :文件IDS

**接口的描述**：

下载完成之后发出确认下载完成确认事件

### 调用说明

参考示例 type: DOWNLOAD\_NOTICE

status: SUCCESS

downLoadNotice {

eventId: "b64e227b94fc494c93935b5116f44f07"

key: "key"

length: "length"

messageInfo {

fileId: "184703419905347591.mp4"

fileSize: "5360330"

fileName: "haha2.mp4"

}

messageInfo {

fileId: "184703419922124806.mp4"

fileSize: "89022543"

fileName: "179184524343775241.mp4"

}

messageInfo {

fileId: "184703419934707714.mp4"

fileSize: "192011703"

fileName: "\351\237\251\345\233\275\345\245\263\345\233\242.mp4"

}

messageInfo {

fileId: "184703419951484933.mp4"

fileSize: "23630759"

fileName: "\345\221\265\345\221\265.mp4"

}

}

# 通知

## 控制指令通知事件

### 通知说明

##### 通知事件类型： CTRL\_DICTATE\_NOTICE

##### 通知事件名称：CtrlDictateNotice

##### 通知参数

string eventId： 事件id

Dictate dictate： 控制指令(控制指令枚举)

string position 文件播放位置（帧数）

string fileId 文件ID

##### 示例

##### 参考示例：[6.2.1 控制指令通知结构体](#_发送消息通知结构体)

##### 通知描述：

##### 控制中心发送XXX指令的通知至设备终端 ，终端收到播放通知事件后解析到数据进行业务处理。

指令START：开始播放

##### 指令PLAY：终端收到播放指令后,调用自身播放器启动,向服务端发送播放进度事件获取播放进度进行播放

指令STOP:暂停播放

指令CLOSE：关闭播放器

指令END： 结束播放

## 下载通知事件

### 通知说明

##### 通知事件类型： DOWNLOAD\_NOTICE

##### 通知事件名称：DownLoadNotice

##### 通知参数

string eventId： 事件id

string key 秘钥

string length 加密后长度

MessageInfo messageInfo 下载通知内容结构体（repeated多参数）

##### 示例

下载通知参考示例：

type: DOWNLOAD\_NOTICE

status: SUCCESS

downLoadNotice {

eventId: "b64e227b94fc494c93935b5116f44f07"

key: "179184524373135360"

length: "16"

messageInfo {

fileId: "184703419905347591.mp4"

fileSize: "5360330"

fileName: "haha2.mp4"

title:"这是标题"

content:"这是内容"

imageUrl:"这是图片地址"

}

messageInfo {

fileId: "184703419922124806.mp4"

fileSize: "89022543"

fileName: "179184524343775241.mp4"

}

messageInfo {

fileId: "184703419934707714.mp4"

fileSize: "192011703"

fileName: "haha222.mp4"

}

messageInfo {

fileId: "184703419951484933.mp4"

fileSize: "23630759"

fileName: "haha22.mp4"

}

}

##### 通知描述：

##### 控制中心发送下载指令的通知至设备终端 ，终端收到下载通知事件后解析到数据进行业务处理

## 删除通知事件

### 通知说明

##### 通知事件类型： DELETE\_NOTICE

##### 通知事件名称：DeleteNotice

##### 通知参数

string eventId： 事件ID

string fileIds 文件ID(多参数以”，”分割)

##### 示例

##### 参考示例：[6.2.1 控制指令通知结构体](#_发送消息通知结构体)

**通知描述：**

##### 控制中心发送删除指令的通知至设备终端 ，终端收到删除通知事件后解析到数据进行业务处理

## 播放进度通知事件

### 通知说明

##### 通知事件类型： PLAY\_PROGRESS\_NOTICE

##### 通知事件名称：PlayProgressNotice

##### 通知参数

string eventId： 事件ID

string equipmentId： 设备ID

string fileId 文件ID

string position 文件位置

PlayStatus playStatus 播放状态

##### 示例

##### 参考示例：[6.2.1 控制指令通知结构体](#_发送消息通知结构体)

**通知描述：**

**终端发送playProgressRequest事件后 ，**

**如果文件在播放中返回PLAY\_PROGRESS\_NOTICE通知。**

例：type: PLAY\_PROGRESS

status: SUCCESS

PlayProgressNotice{

eventId: "1"

equipmentId: "EQID"

FileId: "FILEID"

position: "66666"

}

**如果文件不在播放中 返回播放进度请求失败**

例：type: PLAY\_PROGRESS

status: FAILURE

playProgressResponse {

eventId: "1"

fileId: ""

position: ""

}

示例：

## 注册请求示例

##### 创建RegisterRequest请求 java示例：

##### RegisterRequest registerRequest= RegisterRequest.newBuilder()

##### setEventId(eventId).build();

##### MessageRequest req=MessageRequest.newBuilder().setType(Types.REGISTER)

##### .setRegisterRequest(registerRequest).build();

##### ctx.writeAndFlush(req);

eventId：事件ID

Types.REGISTER*： 请求事件类型*

## 消息通知示例

### **控制指令通知结构体**

##### type：CTRL\_DICTATE\_NOTICE

##### status：SUCCESS

##### CtrlDictateNotice{

##### eventId ：“1” //事件ID

##### dictate: start //指令类型

##### position:“12121212” //文件播放或者暂停等帧数位置

##### fileId：“fileid” //文件id

##### }