moCloud--数据通信格式

# 历史

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Author** | **Date** | **Description** |
| V1.0.0 | WuJinlei | 20180226 | Create |
| V1.0.1 | WuJinlei | 20180327 | 增加2.3节数据传输的格式； |
|  |  |  |  |

# 目录

[moCloud--数据通信格式 1](#_Toc28752)

[历史 1](#_Toc9090)

[目录 2](#_Toc15940)

[1.概述 3](#_Toc7848)

[2.通信规则 3](#_Toc17010)

[2.1.密钥协商 3](#_Toc17253)

[2.2.控制指令 4](#_Toc2568)

[2.3.数据传输 5](#_Toc28247)

[3.各业务的特殊指令格式定义 5](#_Toc6834)

[3.1.注册、登录、退出登录 5](#_Toc17399)

[3.2.获取文件信息列表 5](#_Toc20120)

[3.3.心跳包返回值 6](#_Toc10007)

# 1.概述

本文档，定义client和server通信时，支持的命令，以及各自所携带的body的格式定义，为client和server的通信定义规则。

# 2.通信规则

Client和server的通信，分为两部分：header和body；

根据不同的阶段，也分为两种：密钥协商的相关通信；控制类的通信；

基本原则是：header是size固定的结构体，都是经过加密了的密文形式传送；body以精简的二进制为主，不执行加密，明文传输；

## 2.1.密钥协商

客户端发起的密钥协商的request header定义如下：

typedef struct

{

char mark[MOCLOUD\_MARK\_MAXLEN]; //"MOCLOUD\_CLIENT"

MOCLOUD\_CMDID cmdId; //MOCLOUD\_CMDID\_KEYAGREE

MOUTILS\_CHECK\_CRCVALUE crc32;

}MOCLOUD\_KEYAGREE\_REQUEST;

密钥协商请求没有body；

服务器在接收到client的connect请求后，就等待密钥协商的request到来。对：

* Mark
* cmdId
* Crc32

都执行相应的检查，确认无误后，进行密钥分配，并发送返回值，返回值格式为：

typedef union

{

char desKey[MOCRYPT\_DES\_KEYLEN];

char des3Key[MOCRYPT\_DES\_KEYLEN \* 3]; //can just use 16bytes

char rc4Key[MOCRYPT\_RC4\_KEY\_MAX\_LEN];

char rsaKey[128]; //128bytes=1024bits, 1024RSA

}MOCLOUD\_CRYPT\_KEY\_INFO;

typedef struct

{

MOCLOUD\_CRYPT\_ALGO cryptAlgoNo;

MOCLOUD\_CRYPT\_KEY\_INFO cryptKey;

int keyLen;

}MOCLOUD\_CRYPT\_INFO;

typedef struct

{

char mark[MOCLOUD\_MARK\_MAXLEN]; //"MOCLOUD\_SERVER"

MOCLOUD\_CMDID cmdId; //MOCLOUD\_CMDID\_KEYAGREE

int ret; //KeyAgree succeed or failed

MOCLOUD\_CRYPT\_INFO info;

MOUTILS\_CHECK\_CRCVALUE crc32;

}MOCLOUD\_KEYAGREE\_RESPONSE;

## 2.2.控制指令

除了密钥协商之外的其他控制指令，使用同一个request header和response header：

typedef struct

{

char mark[MOCLOUD\_MARK\_MAXLEN]; //"MOCLOUD\_CLIENT"

MOCLOUD\_REQUEST\_TYPE isNeedResp;

MOCLOUD\_CMDID cmdId;

int bodyLen; //If donot have body, this set to 0; else, after this request, send a body in this length;

MOUTILS\_CHECK\_CRCVALUE crc32;

}MOCLOUD\_CTRL\_REQUEST;

typedef struct

{

char mark[MOCLOUD\_MARK\_MAXLEN]; //"MOCLOUD\_SERVER"

MOCLOUD\_CMDID cmdId;

int ret; //The ret for this order

int bodyLen;

MOUTILS\_CHECK\_CRCVALUE crc32;

}MOCLOUD\_CTRL\_RESPONSE;

## 2.3.数据传输

客户端向服务器请求下载数据时，先发送控制指令，开始下载。服务器得到该指令后，主动向该客户端的数据端口发送数据。发送数据，同样由header和body组成，为了效率考虑，暂时不进行加密。

Header的格式为：

typedef struct

{

char mark[MOCLOUD\_MARK\_MAXLEN]; //MOCLOUD\_SERVER

int fileId; //use this id, client can assure which file it is, instead of fileKey

int unitId;

int bodyLen; //The length, except the last unit, bodyLen is the unitLen

unsigned char checkSum;

}MOCLOUD\_DATA\_HEADER;

Mark用来标记这个header的开始；

Fileid是要下载的文件id，这个id可以与要下载的文件一一对应，由client自行维护“<id, file>”的映射关系；

unitId是这次读取的unit的id。传输采用unit为基本单位进行，每个unit的长度固定为N字节，最后一个unit如果不足N字节，以实际长度为准。写入文件时，根据该unitId计算本部分数据应写入的位置。

bodyLen，只有最后一个unit的长度可能小于等于N字节，其他unit的长度，都恒为N；

checkSum，用来做校验。

Header后紧跟bodyLen个字节的body；

# 3.各业务的特殊指令格式定义

客户端和服务器通信时，可以通过body来进行额外数据的发送，本章节介绍各个场合下的通信数据的组成格式；

## 3.1.注册、登录、退出登录

注册、登录、退出登录，都是首先发送一个标准的request header(密文形式)，之后尾随一个如下格式的body(明文格式)：username=%s, password=%s；

解析规则是：第一个username和最后一个“,password=”之间的所有内容，都是用户的username；最后一个“,password=”之后的所有内容，都是用户的password；

## 3.2.获取文件信息列表

获取文件信息列表，是由client主动发起，server返回要查询的类型的所有文件的基本信息的操作；

Client发起请求时，需要告知server要获取的是哪一种类型的文件列表，**这个type，是占用了MOCLOUD\_CTRL\_REQUEST->bodyLen这个属性来实现的**；

Server的返回值，除response header(加过密的密文)外，还会附带一个response body(明文)；该response body的组成，就是MOCLOUD\_BASIC\_FILEINFO这个二进制数据的依次排列；

Client解析时，按照顺序，通过sizeof(MOCLOUD\_BASIC\_FILEINFO)可以获知每个文件的信息；

## 3.3.心跳包返回值

Server返回给client的心跳包的返回值里:

* MOCLOUD\_CTRL\_RESPONSE->ret值为MOCLOUD\_HEARTBEAT\_RET\_FAILED，说明server处理该client的心跳包失败了；为MOCLOUD\_HEARTBEAT\_RET\_OK说明处理成功了，而且server的文件列表没有变更；为MOCLOUD\_HEARTBEAT\_RET\_FILELIST\_CHANGED说明文件列表变化了，用户需要更新这个特定类型的文件列表；
* MOCLOUD\_CTRL\_RESPONSE->bodyLen，在上述ret为MOCLOUD\_HEARTBEAT\_RET\_FILELIST\_CHANGED的情况下，该bodyLen代表了发生变化的文件的类型，client可以根据该类型，只更新这一种类型的文件信息；