# Protobuf

协议头

|  |  |
| --- | --- |
| 意义 | 数据类型 |
| 指令类型 | char |
| 指令号 | char |
| 序列号 | int32 |
| 版本号 | char |
| 包体长度 | int32 |

连接协议 （控制套接字）

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 指令类型 | 指令号 | 内容 |
| 请求连接 | 1 | 1 |  |
| 退出连接 | 1 | 2 |  |
| 请求连接回复 | 1 | 100 | {  resultType:“int32”  //1成功，-100失败  failReason:“string”  colorPort:“int32”//端口号  depthPort:”int32”  skelePort:”int32”  } |
| 请求设备列表 | 1 | 3 |  |
| 回复设备列表 | 1 | 101 | {  deviceList: “string”  Separator: “string”  } |

骨骼,三维点云协议（控制套接字）

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 指令类型 | 指令号 | 内容 |
| 开始请求 | 2 | 1 | {  deviceName:”string”  } |
| 回复开始请求 | 2 | 100 | {  deviceName:”string”  guid:”string”  resultType:“int32”  //1成功，-100失败  failReason:“string”  } |
| 服务器主动结束 | 2 | 101 | {  deviceName:”string”  reason:”string” // 断开连接的任意附加字符串信息  }备注：设备断开后将发生此事件 |
| 结束请求 | 2 | 2 | {  deviceName:”string”  reason:”string”// 断开连接的任意附加字符串信息  } |

备注:

1. 一个客户端仅对应一个控制套接字，此外客户端是个单例，也就是每台PC上只允许同时运行一个客户端程序
2. 每个控制套接字可以请求多个设备的数据，但不允许同时请求同一设备的数据两次或多次。亦就是说，如果一个客户端成功请求了某一设备的数据，即不能再次请求该设备的数据，直到该设备的数据传输套接字全部断开。即使由于未知错误发生这种重复请求，该连接请求也会被服务器忽略，不予受理。
3. 客户端向服务器结束请求时发送设备名字，服务器仅断开该设备的所有数据套接字（颜色、深度、骨骼），而不会影响该客户端与服务器的其他设备的数据传输。
4. 在控制套接字上，客户端主动请求退出连接或者未知错误断开了与服务器的连接，服务器会断开该客户端所有设备的数据套接字连接。
5. GUID由服务器产生，客户端的数据套接字返回该GUID。服务器端用以关联控制套接字 和 该控制套接字对应的所有数据套接字。

# Kinect数据协议

**客户端 到 服务端（仅在建立连接后发送一次）:**

协议头（和控制套接字用的是同一类型的头部）

|  |  |
| --- | --- |
| 意义 | 数据类型 |
| 指令类型 | char |
| 指令号 | char |
| 序列号 | int32 |
| 版本号 | char |
| 包体长度 | int32 |

数据信道 （数据套接字）

|  |  |  |  |
| --- | --- | --- | --- |
| 功能 | 指令类型 | 指令号 | 内容 |
| 由数据传输socket发送到服务器端(用于关联控制信道和数据信道套接字) | 3 | 1 | {  deviceName:”string”  guid:”string”  } |

**服务端 到 客户端:**

头部

|  |  |
| --- | --- |
| **意义** | **数据类型** |
| 帧编号 | uint32 |
| 数据类型 | uint32 |
| 包体长度 | uint32 |
| 数据（包体） | |

包体

H264/骨骼点