**前端页面与后台服务通信协议**

与服务程序的通信全部使用WebSocket技术, JSON数据交换。

1. 约定

本文档所指定的都是以URL地址

ws://localhost:***19870***/***xxx***

其中端口地址,后期可能会换掉,xxx是文档中提高的路径。

文档当前是按照内部完整模式来列功能的。

有一些常用的JSON属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| SEQ | 前端发送JSON命令时附带的一个随机字符串，服务在返回的时候会原样附带 |
| CMD | 前端请求对应数据的操作 |
| ERROR | SUCCESS, FAILURE, TIMEOUT |
| ERROR\_MESSAGE | 在错误的基础上，增加的解释说明 |
| EVENT | 服务端主动发送的一段通知性的数据。对应成各种通知事件。 |
| FILE | 每个可以绑定的设备都会有一个唯一的文件名。一个设备可以提供多个服务接口 |
|  |  |

从服务端发来的有两种消息，一种是EVENT，一种是应答，

即同一时间，EVENT与SEQ只会存在一个。

1. 前端框架要求

前端会使用WebKit或者类型的核心写一个浏览器框架，主要是为了保护在浏览器里执行的代码。

如果浏览器无法保证JS代码的安全，那么会自己写一个浏览器框架。

1. ROOT服务

路径

/general

除了基础服务。

比如版发现的设备版本为

2.2.2

对应的FILE的base64编码为

SfjisdleasdIEFef3r

如果服务路径为

/config

那么真实服务路径为

/config/SfjisdleasdIEFef3r

其中服务路径的附加部分是base64编码，本质上仍然是json数据。

这样简化连接成功过程的交互次数。

* 1. 设备发现协议

每个接口，都会提供设备的增加与删除信息。

* 1. 设备信息格式

|  |  |
| --- | --- |
| **字段** | **说明** |
| **product\_type** | 产品类型，比如A2，A3，Inspire1 |
| **device\_type** | 设备类型，比如主控，遥控器，电台 |
| **version** | 版本，当前设备的版本。 |
|  |  |

FILE对象，这里文件名发生了很大的变化。

FILE是一个隐藏对象。

{

“FILE\_NAME”: “COM3”, // USB-VID-897ewf

“VERSION”: “2.2.2.2”,

}

以上数据需要经过base64编码后，在连接的路径上附带。

* + 1. Device Arrival

{

“EVENT”: “device\_arrival”,

“PRODUCT\_TYPE”: “A3”,

“DEVICE\_TYPE”: “Controller”,

“VERSION”: “2.2.1.16”,

“FILE”: “COM3”

}

* + 1. Device Removed

{

“EVENT”: “device\_removed”,

“FILE”: “COM3”

}

* 1. 登录

{

“SEQ”: “asdf”,

“CMD”: “login”,

“email”: “[itolfo.pan@dji.com](mailto:itolfo.pan@dji.com)”,

“password”: “asdf;lkj”,

“save\_password”: true,

}

升级的接口需要登录。

1. A3 调参接口
   1. URL地址

/controller/config/user

* 1. 接口格式约定

JSON通信格式，可能出现的字段

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **说明** |
| ***SEQ*** | String | Web端与App程序之间用于确定命令的归属，随机字符串 |
| ***CMD*** | String | WebApp命令。***read, write, reset，book，bind\_device*** |
| ***INDEX*** | String | 要访问的变量名 |
| ***VALUE*** | String | 要设置的变量值或者变量的返回值 |
| ***ERROR*** | String | JSON命令执行结果，FAILURE，SUCCESS，TIMEOUT |
| ***ERROR\_MESSAGE*** | String | 如果出现FAILURE，那么这个会返回说明错误的原因 |
| ***PERIOD*** | Number | 推送数据的发送间隔 |
| ***FILE*** | String | 要绑定的设备名 |

通用事件接口

device\_arrival

device\_removed

* 1. 命令说明
     1. 通知类消息

连接设备成功后，如果一切正常，会首先推送一段进度数据。然后告诉前端，服务当前就绪了，可以正常的调整数据了。

* + - 1. sync\_progress

连接成功后，后台服务可能会花费一定的时间来同步数据，这时候会推送同步的进度。

{

“EVENT”: “sync\_progress”,

“PROGRESS”: 99

}

* + - 1. initial\_ready

WebSocket连接成功后，并不代表可以进行数据通信了，这时候后台服务可能还需要同步数据等等操作，这时候会推送上面的消息，直到所有的操作都OK后，会推送当前消息，这时候，就可以正常的操作了。

初始化后的变量范围列表。

{

“EVENT”: “initial\_ready”,

“LIST”:

{

“imu\_x”: {“MIN”: 2.1, “MAX”: 9.9},

“imu\_y”: {“MIN”: 2.1, “MAX”: 9.9},

“imu\_z”: {“MIN”: 2.1, “MAX”: 9.9},

…

“imu\_g”: {“MIN”: 2.1, “MAX”: 9.9}

}

}

* + - 1. data\_update

数据更新。当前端定义了数据的组的发送时间后，后台会定时通过data\_update发送对应数据的更新。

{

“EVENT“: “data\_update”,

“INDEX”: “imu\_x”,

“VALUE”: 5.6

}

每个数值，只要发生了改变，都会通过data\_update发送到每一个已连接的WebSocket

* + 1. 参数操作接口
       1. read

发送给WebSocket的消息

{

“SEQ”: “rasdfsf”,

“CMD”: “read”,

“INDEX”: “usr\_sss\_sss\_uuu”

}

应答消息

{

“SEQ”: “rasdfsf”,

“ERROR”: “SUCCESS”,

“VALUE”: 2.3

}

这里，如果成功，那么返回的是新的数据，如果失败，那么这个Value不会存在。

* + - 1. write

{

“SEQ”: “rasdfsf”,

“CMD”: “write”,

“INDEX”: “usr\_sss\_sss\_uuu”,

“VALUE”: “435.33”

}

ACK

{

“SEQ”: “rasdfsf”,

“ERROR”: “SUCCESS”,

“VALUE”: “435.33”

}

注意，Write成功后，不仅会有一个ACK，还会把成功的值通过data\_update事件推送给当前所有已连接的WebSocket

* + - 1. reset

{

“SEQ”: “rasdfsf”,

“CMD”: “reset”,

“INDEX”: “usr\_sss\_sss\_uuu”

}

reset也会与Write有相同的效果，即reset成功后，还会发生一次data\_update事件。

* + - 1. book

{

“SEQ”: “rasdfsf”,

“CMD”: “book”,

“INDEX”: “usr\_sss\_sss\_uuu”

}

book所产生的结果数据，都是通过data\_update发送出来。

1. 升级

URL

/controller/upgrade

一键升级的逻辑是由前端完成的。

前端脚本一个一个的给所有的设备升级

* 1. 刷新设备状态，设备最新版本信息

{

“SEQ”: “asdfasdfasdf”,

“CMD”: “refresh\_device\_status”

}

* 1. device\_info

{

“EVENT”: “device\_info”

“DEVICE”: “imu”,

“DEVICE\_INDEX”: “0”

“LOADER”: “2.3.4.5”,

“FIRMWARE”: “3.4.3.4”,

“LATEST\_FIRMWARE”: “3.4.3.5”

“LATEST\_DETAIL”: “http://234234/234234/23/4”

}

同一个设备，会有好几个子设备。

比如A3，连接上后，会依次推送其连接上的子设备的信息。

Loader的升级暂时并不开放。

A3的飞控系统最小有如下设备

controller,

imu,

gps,

imu pro,

* 1. 进入升级
     1. upgrade

{

“SEQ”: “asfsadf”

“CMD”: “upgrade”,

“DEVICE”: “imu”

“DEVICE\_INDEX”: 0

}

* 1. 升级进度

给每个设备升级，都会有以下几个状态，直到返回finish，设备升级才算成功。

这时候，可以给另一个设备升级。

{

“EVENT”: “upgrade\_step”

“UPGRADE\_STEP”: “enter\_loader”, prepare, data\_transmission, verify, reboot, finish

“DEVICE”: “imu”,

“DEVICE\_INDEX”: 23

“PROGRESS”: optional, only data\_transimission

}

* 1. BETA升级

用户可以尝试最新的固件，或者给特定用户的固件。需要用输入用户名及密码。

在前端页面增加一个BETA的按钮，让用户输入用户名及密码。

{

“CMD”: “beta\_upgrade\_query”, “beta\_upgrade”

“DEVICE”: “imu”,

“DEVICE\_INDEX”: 4,

“USER”: “asdfsadf”

“PASS”: “asdfsadf”

}

查询后，返回信息与device\_info相同。

* 1. 内部调试命令
     1. upgrade\_file

{

“SEQ”: “asfsadf”

“CMD”: “upgrade\_file”,

“DEVICE”: 3,

“DEVCIE\_INDEX”: 0, // 0 is default, user can input

“DEVICE\_FIRMWARE\_TYPE”: 0, // 0 is default

“FILE\_DATA”: “5566ccdde” // base64

}

FILE\_DATA是base64后的数据。

1. 飞行模拟器（TBD）

模拟器，当前支持的是内部模拟器，即主控内部的设置。

JSON格式

1. 地面站
   1. navigation

/controller/navigation

* + 1. 接口格式约定

JSON通信格式，可能出现的字段

|  |  |  |
| --- | --- | --- |
| **字段名称** | **字段类型** | **说明** |
| ***SEQ*** | String | Web端与App程序之间用于确定命令的归属，随机字符串 |
| ***OPERATION*** | String | WebApp命令。 |
| ***VALUE*** | String | 要设置的变量值或者变量的返回值 |
| ***ERROR*** | String | JSON命令执行结果，FAILURE，SUCCESS，TIMEOUT |
| ***ERROR\_MESSAGE*** | String | 如果出现FAILURE，那么这个会返回说明错误的原因 |
| ***EVENT*** | String | 通知类事件，如数据推送data\_update |

* + 1. 命令说明
       1. SetNavigationMode

{

“SEQ”: “sdfs211”

“OPERATION”: “SetNavigationMode”

“VALUE”: {

“NAVIGATION\_MODE”: 1 // 1: Enter 2: Exit

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

}

* + - 1. UploadWayLineMission

{

“SEQ”: “sdfs211”

“OPERATION”: “UploadWayLineMission”

“VALUE”: {

“LENGTH”: 5

“VEL\_CMD\_RANGE”: 5

“IDLE\_VEL”: 3

“ACTION\_ON\_FINISH”: 0

“MISSION\_EXEC\_NUM”: 5

“YAW\_MODE”: 0

“TRACE\_MODE”: 0

“ACTION\_ON\_RC\_LOST”: 0

“GIMBAL\_PITCH\_MODE”: 0

“HP\_LATI”: 0

“HP\_LONGTI”: 0

“HP\_ALTI”: 200

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

}

* + - 1. DownloadWayLineMission

{

“SEQ”: “sdfs211”

“OPERATION”: “DownloadWayLineMission”

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

“VALUE”: {

“LENGTH”: 5

“VEL\_CMD\_RANGE”: 5

“IDLE\_VEL”: 3

“ACTION\_ON\_FINISH”: 0

“MISSION\_EXEC\_NUM”: 5

“YAW\_MODE”: 0

“TRACE\_MODE”: 0

“ACTION\_ON\_RC\_LOST”: 0

“GIMBAL\_PITCH\_MODE”: 0

“HP\_LATI”: 0

“HP\_LONGTI”: 0

“HP\_ALTI”: 200

}

}

* + - 1. UploadWayPoint

{

“SEQ”: “sdfs211”

“OPERATION”: “UploadWayPoint”

“VALUE”: {

“INDEX”: 0

“LATI”: 0

“LONTI”: 0

“ALTI”: 120

“DAMPING\_DIS”: 180

“TGT\_YAW”: 30

“TGT\_GIMBAL\_PITCH”: -120

“TURN\_MODE”: 0

“HAS\_ACTION”: 0

“ACTION\_TIME\_LIMIT”: 50

“ACTION”: {

“ACTION\_NUM”: 15

“ACTION\_RPT”: 15

“COMMAND\_LIST”:{

“WP\_ACTION\_STAY”: 200

.

.

.

}

}

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

“VALUE”:{

“INDEX”: 0

}

}

* + - 1. DownloadWayPoint

{

“SEQ”: “sdfs211”

“OPERATION”: “DownloadWayPoint”

“VALUE”:{

“INDEX”: 2

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

“VALUE”: {

“INDEX”: 2

“LATI”: 0

“LONTI”: 0

“ALTI”: 120

“DAMPING\_DIS”: 180

“TGT\_YAW”: 30

“TGT\_GIMBAL\_PITCH”: -120

“TURN\_MODE”: 0

“HAS\_ACTION”: 0

“ACTION\_TIME\_LIMIT”: 50

“ACTION”: {

“ACTION\_NUM”: 15

“ACTION\_RPT”: 15

“COMMAND\_LIST”:{

“WP\_ACTION\_STAY”: 200

.

.

.

}

}

}

}

* + - 1. StartOrCancelWayLineMission

{

“SEQ”: “sdfs211”

“OPERATION”: “StartOrCancelWayLineMission”

“VALUE”: {

“GO\_STOP”: 0 // 0: go 1: stop

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

}

* + - 1. PauseOrContinueWayLineMission

{

“SEQ”: “sdfs211”

“OPERATION”: “PauseOrContinueWayLineMission”

“VALUE”: {

“PAUSE”: 0 // 0: pause 1: continue

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

}

* + - 1. SetWayLineFlightIdelValue

{

“SEQ”: “sdfs211”

“OPERATION”: “SetWayLineFlightIdelValue”

“VALUE”: {

“IDLE\_VEL”: 0

}

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

“VALUE”: {

“IDLE\_VEL”: 0

}

}

* + - 1. GetWayLineFlightIdelValue

{

“SEQ”: “sdfs211”

“OPERATION”: “GetWayLineFlightIdelValue”

}

ACK

{

“SEQ”: “sdfs211”

“ERROR”: “SUCCESS”

“VALUE”: {

“IDLE\_VEL”: 0

}

}

* + - 1. PushNavigationStatusInfo

{

“EVENT“: “data\_update”,

“OPERATION”: “PushNavigationStatusInfo”,

“VALUE”: {

“MISSION\_TYPE”: 0

“TARGET\_WAYPOINT”: 10

“CURR\_STATE”: 0

“ERROR\_NOTIFICATION”: 0

}

}

OR

{

“EVENT“: “data\_update”,

“OPERATION”: “PushNavigationStatusInfo”,

“VALUE”: {

“MISSION\_TYPE”: 1

“LAST\_MISSION\_TYPE”: 10

“IS\_BROKEN”: 0

“REASON”: 0

}

}

* + - 1. PushNavigationEventInfo

{

“EVENT“: “data\_update”,

“OPERATION”: “PushNavigationEventInfo”,

“VALUE”: {

“INCIDENT\_TYPE”: 0

“WAYPOINT\_INDEX”: 0

“CURR\_STATE”: 0

}

}

OR

{

“EVENT“: “data\_update”,

“OPERATION”: “PushNavigationEventInfo”,

“VALUE”: {

“INCIDENT\_TYPE”: 1

“REPEAT”: 0

}

}

* 1. 下载JavaScript应用

支持第三方应用。航点，云台，相机控制开放后。第三方开发者可以根据JS API内容自行定制App应用。

配合App或者JS应用。

1. 数据上传分析

服务端支持扫描本地可移动的U盘。然后自动读取数据，分析回放飞行过程。

1. 通用指令（工厂临时应用，调试）

通用指令,内部专有协议的Web翻译。正常情况下,Web发送的都是简化过的。完整版本的控制由服务端封装。

通用命令目前仅限内部调试使用。

一次JSON通信只能发送一条命令。队列式的命令。

发送JSON

{

“SEQ”: ”afsefaef”,

“CMD”: “raw”,

“CMD\_ID”: 0x87,

“CMD\_SET”: 0,

“SENDER\_TYPE”: 3,

“SENDER\_ID”: 0,

“RECEIVER\_TYPE”: 4,

“RECEIVER\_ID” : 0,

“DATA” : “0XFF”

}

这里发送的命令需要在前端使用JS组成数据包。前端需要知道每个原始的协议格式。

应答数据里，会恒定有一个ERROR字段，用来表示命令是否成功，超时，或者发生的错误。

返回JSON

{

“SEQ”: ”afsefaef”,

“ERROR”: “SUCCESS”,

“DATA” : “0XFF”

}