|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 主动发起方 | 触发条件 | 服务器端接口 | 客户端请求格式 | 数据返回格式 |
| 客户端or服务端 | 定时，等待列表底限 |  | 1 |  |
| 客户端 | 定时 |  | 2 |  |
| 客户端 |  |  | 3 |  |
| 客户端 |  |  | 4 |  |
| 客户端or服务端 |  |  | 5 |  |
| 客户端or服务端 |  |  | 6 |  |

服务器，客户端接口通信文档

客户端，服务器交互示意图表

图表数字和符号的含义和下文一致

服务器端包含的所有接口：

|  |  |  |
| --- | --- | --- |
| 服务器接口名称 | Action指令名称 | 请求方向 |
| 1. 更新任务列表 | update\_task\_list | 客户端 服务器端 |
| 1. 批量更新当前任务状态 | update\_much\_taskinfo | 客户端 服务器端 |
| 1. 客户端回报数据 | upload\_client\_data | 客户端 服务器端 |
| 1. 客户端上传状态 | upload\_client\_status | 客户端 服务器端 |
| 1. 更新代理数据 | update\_proxy \_data | 客户端 服务器端 |
| 1. 更新cookie数据 | update\_cookie \_data | 客户端 服务器端 |

关于①：

希望获取的任务类型和数量，服务器端不一定满足，只是个建议，服务器可以有自己针对这种终端或这个终端的配置，或算法

关于②：

每个任务都有属性，记录 是否有变化

定时收集有变化的任务id，id放入等待上传列表里，属性变为没变化

如果上传成功，则从队列里移除

关于③：

多台Client端上传时，服务器只接受当前服务器认可分配的Client的数据

其他的回报建议删除

客户端请求服务器端的接口数据格式

（客户端的数据格式标号对应上面服务器端的接口标号）

说明：上传的数据格式一致，服务器端根据不同的action进行相应的解析

数据格式的的表格定义：

一：整体表格代表客户端的整体请求协议

二：左1：代表数据的字段，作为key

三：左2：代表每个字段所要携带的必要信息，作为value

客户端数据格式图表

|  |  |  |  |
| --- | --- | --- | --- |
| device | type（客户端类型） | pc | |
| id(此类型中唯一) |  | |
| mac (选填) |  | |
| api\_key(选填) |  | |
| Other |  | |
| command | action(执行的动作) | update\_task\_list | |
| version(版本) |  | |
| body | taskstats(任务简报) | time |  |
| status |  |
| get\_task（得到需要类型的任务和数量） | Type | 数量 |
| Type1 |  |
| Task s 在Action为更新状态时有意义 | tasks | Id1 | 字段：更改的数据 |
| Id2 |

具体的json格式：

{

“device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:"update\_task\_list", #任务类型

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: ／／简报具体内容

[

{"type":"type1","count":200,"wait":100,"run":50,"comple",2000,“effc":201.5}

任务类型，总数，等待执行数，正在运行数，自上次报告后完成次数，执行效率（次／秒）

{"type":"type2","count":200,"wait":100,"run":50,"comple",2000,“effc":201.5}

]

｝，

//希望获取的任务类型和数量，服务器端不一定满足，只是个建议，服务器可以有自己针对这种终端或这个终端的配置，或算法

“get\_tasks”：

｛“type1"：10000，

"type2"：2000｝，

tasks{id:{

字段1：更新的数值，字段2：更新的数值

}}

}

注意：以下操作都是在上边数据图表的基础上进行

（下面的json数据中的字段为空表示服务器不关心此字段的信息）

1. 更新服务器端的服务列表(action: update\_task\_list)

具体的json数据格式如下：

device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:"update\_task\_list", #更新任务列表

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: ／／简报具体内容

[

{"type":"type1","count":200,"wait":100,"run":50,"comple",2000,“effc":201.5}

任务类型，总数，等待执行数，正在运行数，自上次报告后完成次数，执行效率（次／秒）

{"type":"type2","count":200,"wait":100,"run":50,"comple",2000,“effc":201.5}

]

｝，

//希望获取的任务类型和数量，服务器端不一定满足，只是个建议，服务器可以有自己针对这种终端或这个终端的配置，或算法

“get\_tasks”：［

｛“type1"：10000｝，

｛ "type2"：2000｝，

］

tasks:{}

data:{}

}

1. 批量更改当前任务状态（action: update\_much\_taskinfo）

device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:"update\_much \_taskinfo ", #批量更改当前任务状态

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: []／／简报具体内容

｝，

“get\_tasks”：［

］

tasks{id:{

字段1：更新的数值，字段2：更新的数值

}}

data:{}

}

1. 客户端回报数据(action:upload\_client\_data)

device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:" upload\_client\_data ", #客户端回报数据

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: []／／简报具体内容

｝，

“get\_tasks”：［

］

tasks{}

data:{data\_id:[format,data],data\_id2: [format,data]}

data\_id为数据的唯一编号，上传数据任务编号根据客户端id和时间

产生数据包的id

Format代表是否压缩，是否加密

Data是回报的数据

}

1. 客户端状态上传(action:upload\_client\_status)

device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:" upload\_client\_status ", #客户端状态上传

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: []／／简报具体内容

｝，

“get\_tasks”：［

］

tasks{}

data:{ }

‘client\_status’：‘0‘or’1’//服务端状态0代表异常，1代表正常

}

1. 更新代理数据（action: update\_proxy \_data）

device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:" update\_proxy \_data ", #更新代理数据

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: []／／简报具体内容

｝，

“get\_tasks”：［

］

Task:{}

Data:{ }

‘Client\_status’：//服务端状态0代表异常，1代表正常

‘proxy\_data’:[{‘ip’:….,’port’:…..,’type’:…,’url’:…..},{ ‘ip1’:….,’port1’:…..,’type1’:…,’url1’:…..}]//代理数据 type指http/https，url:指适用于的平台

}

1. 更新cookie数据（action: update\_cookie \_data）

device": //客户端信息

{

"type":"pc", 暂时只有PC

"id":"…..", 在此类型中必须唯一

--------------------以上必选---------------

"nankename":" "

"mac":"……."

API\_KEY

}

"command":

{ action:" update\_proxy \_data ", #更新代理数据

"version":"1.2.3"

}

"body":{

" taskstats"： //平台任务简报 客户端－－－》服务器

｛

"time"：浮点数， 自1970.1.1起的描述，格林威治 local时间，简报时间

status: []／／简报具体内容

｝，

“get\_tasks”：［

］

Task:{}

Data:{ }

‘Client\_status’://服务端状态0代表异常，1代表正常

‘proxy\_data’:

cookie\_data’:{‘jd’:[{‘sid’:……..,’\_jdu’:……..,’\_jdv’:………..,’\_jda’:},],’tianmao’:[]} //京东cookie数据，每个平台需要的cookie数据不同。

}

对服务器的所有接口操作都会返回统一的数据，数据格式如下：

{

"status":1/0,#表示执行成功还是失败

"error":"错误文档"

"content":内容

}

1. ：服务器端①返回请求数据1的数据格式如下：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Status |  | | | |
| error |  | | | |
| content | action | Add/update/delete | | |
| task | command | type |  |
| version |  |
| Id(业务编号) |  |
| Guid（任务唯一编号） |  | |
| delay |  | |
| timeout |  | |
| body |  | | |

Content可以有多个返回的任务，任务包括增加任务，删除任务和更新任务，客户端根据不同的action，从对应的字段拿到需要的数据或进行相应的操作。

具体的json格式如下：

{

“status”:1/0 表示操作的成功／失败

“error”:错误信息

“content”:[

{“action”:add,//对客户端的任务列表操作：增加任务

”task”:{

“command”:{“type”: 任务类型，“version”:版本号，“id”:业务编号}，

“guid”:任务唯一编号，

“delay”: 0表示尽快执行，其他数值表示收到任务后多久开始执行

“timeout”: 表示到达预计执行时间后多长时间后得到结果，超出后算超时

“body”:{

/任务特有信息

//如果数据特别大，可以规定数据下载的方法，索引

}

}

},

{

{“action”: /delete,//对客户端的任务列表操作：删除任务

”task”:{

“command”:{“type”: 任务类型，“version”:版本号，“id”:业务编号}，

“guid”:任务唯一编号，

“delay”: 0表示尽快执行，其他数值表示收到任务后多久开始执行

“timeout”: 表示到达预计执行时间后多长时间后得到结果，超出后算超时

“body”:{

/任务特有信息

//如果数据特别大，可以规定数据下载的方法，索引

}

}

}

}，

{“action”: update,//对客户端的任务列表操作：更新任务

”task”:{

“command”:{“type”: 任务类型，“version”:版本号，“id”:业务编号}，

“guid”:任务唯一编号，

“delay”: 0表示尽快执行，其他数值表示收到任务后多久开始执行

“timeout”: 表示到达预计执行时间后多长时间后得到结果，超出后算超时

“body”:{

/任务特有信息

//如果数据特别大，可以规定数据下载的方法，索引

}

}

},

]

待补充和完善。