1. SSID合法性检查 ----- 找出信号，用户连接上但不做判断

1.1未关联（SSID=null）：

app上传ssidList到app后台（截取一个样例数据）

{ssid bssid，信号强度，信号频率，信道}

判断ssidList是否为空

为空，结束，返回给app当前位置无合法无线

不为空：

调用接口绿洲接口getservicetemplate

请求示例

{"shopId":"4", "devSN":"210235A1JNB161000029"} ----app获取经纬度信息自动获取列表

返回示例

{

"code":0,

"data":{

"Template":[

{

"\_id":"5d8eb8c36defbdb74153775d",

"Name":"sahz",

"devSN":"210235A1JNB161000029",

"shopId":"4",

"Enable":"true",

"SSID":"sahz",

"DefaultVlan":"91",

"HideSsid":"false",

"MaxClientCnt":"0",

"ForwardingLocation":"2",

"PolicyForwarding":"false",

"AssociationLocation":"2",

"AuthenticationLocation":"2",

"QuikAssociation":"false",

"UserIsolation":"false",

"Description":"seeranalyzer hangzhou ac",

"Location":"default-location",

"smoothFlag":"no",

"AkmM=ode":"2",

"bindApCount":6

}

。。。。。。。。。

]

},

"message":"success"

}

处理逻辑：判断ssidList中的ssid是否在接口返回的template中

是：合法的ssid

不是：不合法的ssid

如果ssidlist里没有合法的ssid，返回给app当前位置无合法无线

如果ssidlist里有合法的ssid，继续

判断BSSID是否合法：对上一步所有的合法ssid及其BSSID依次调用绿洲接口getAPInfoByBssid

请求示例

{"BSSID":"b0f9-63e6-8749", "shopId":"4"}

返回参数说明

输出，查询到的AP的信息

返回示例

有数据时：

{

"code": 0, //接口交互成功，0表示成功，其它表示失败

"message": null, //交互失败详细信息

"data":

{

"apSN":"1234567890ABCD"

}

}

查询无数据时返回如下：

无数据时：

{

"code": 0, //接口交互成功，0表示成功，其它表示失败

"message": null, //交互失败详细信息

"data": null,

}

如果查询到的apSN=null，bssid不合法

如果查询到的apSN!=null，bssid合法

如果bssid有非法，提示周围有钓鱼ap（存下告警信息，下面选最强信号时，非法BSSID不会被选到）

如果bssid没有非法，继续

找出最强合法（合法ssid合法bssid）信号（不需要调用接口）：

对同一合法ssid下不同bssid根据信号强度从强到弱排序

找出每个ssid下信号最强的（5G优先，2.4G次之）

后台将找到的信号返回给APP（现在会对每一个合法SSID都找一个合法信号最强的推荐）

（完）

1.2已关联（ssid！=null）： ------多次检查

（出现在每一个testFailStage检测之前，和第二次调用的最前面）

APP上传所有数据到APP后台

（）

首先判断关联的ssid是否合法，调用接口getservicetemplate

关联的ssid不合法，转到1.1

关联的ssid合法，继续

判断关联的ssid的bssid是否合法，调用接口getAPInfoByBSSID

关联的ssid的bssid不合法，当前连接为钓鱼ap（存下告警信息），转到1.1，等1.1结束一起返回给APP（APP会显示告警等信息）

（属性）----判断型号强度并决定是否切换

判断当前连接信号和ssid列表里的信号强度是否要执行切换操作（不需要上传数据）：

判断逻辑：（切换不切换即为返回过程）

1. 当前连接为5g

信号强度大于-55db，不切换

-75db<信**号强度<-55db，最佳5g信号比当前大10db，切换**

信号强度<-75db，最佳5g/2.4g>-70db，切换

1. 当前关联为2.4g

最佳5g>-65db，切换

关联信号>-60db，不切换

关联信号<-75db，最佳2.4g比当前信号强度大10db，切换

1. 接入认证

当APP监控到字段staAssoc=0（wifi接入认证时长）时，上传数据到APP后台

（数据）

APP后台调用绿洲接口getStaAuthStatus

入参为

{

“MAC”: “1a2b-3c4d-5e6f”,

“startTime”: “2019-08-15 16:00:00”, ------自定义-时间

“endTime”: “2019-08-15 17:05:01”, --------用户接入时间

“limit”:2,

“type”:3

}

返回为(data中“PSK”“MAC”“802.1x”可以全有，全没有，部分有)

* 1. 有数据时：
  2. {
  3. "code": 0, //接口交互成功，0表示成功，其它表示失败
  4. "message": null, //交互失败详细信息
  5. "data":
  6. [
  7. {
  8. "PSK": “success” //最新一条数据在最前面，以此类推
  9. “failReason”:1 //失败原因
  10. },
  11. {
  12. "MAC": “fail” ----------------------失败，none代表没要求
  13. “failReason”:1 //失败原因
  14. }
  15. {
  16. "802.1x": “fail”
  17. “failReason”:1 //失败原因
  18. }
  19. ]
  20. }
  21. 无数据时：
  22. {
  23. "code": 0, //接口交互成功，0表示成功，其它表示失败
  24. "message": null, //交互失败详细信息
  25. "data": null,}

查看返回值中data中PSK，MAC，802.1x字段的状态（success，failed）判断认证成功/失败。

1. DHCP失败检测

当ipDHCP=0时，上传数据到APP后台

（数据）

后台调用云数接口getEventList

入参为

pageNum=1

pageSize=5

startTime=1565682930495

endTime=1565769330495

mac=30:63:6b:9b:f4:5f

type= OASIS\_DHCP

返回为

var dhcp={

    "data": {

        "total": 4,

        "list": [{

            "child": [{

                "title": "DHCP ACK",

                "describe": "DHCP服务器发送ACK应答报文，通知用户可以使用分配的IP地址",

                "eventId": "156576600000094:87:e0:33:57:43",

                "eventType": "OASIS\_DHCP",

                "time": "2019-08-14T15:00:00.471+08:00",

                "objectMac": "94:87:e0:33:57:43",

                "objectIp": "",

                "host": "219801A0TB9168Q00583",

            }],

            "eventId": "156576600000094:87:e0:33:57:43",

            "describe": "",

            "title": "DHCP申请地址消息",

            "startTime": "2019-08-14T15:00:00.471+08:00",

            "eventType": "OASIS\_DHCP",

            "endTime": "2019-08-14T15:00:00.001+08:00",

            "content": "{\"OP\":3,\"timeConsuming\":\"471\",\"msgState\":3,\"stateDescribe\":\"DHCP ACK超时\",\"SN\":\"序列号：219801A0TB9168Q00583\",\"type\":\"DHA\",\"pointTime\":1565766000471}"                }]

    }

判断content中stateDescribe字段是不是DHCP ACK成功。

1. portal认证

当字段GWPingLost=100%时，app上传数据到APP后台 ------判断逻辑

（数据）

后台调用云数接口getAuthInfoByMac

请求示例

{

“staMAC”: “1a：2b：3c：4d：5e：6f”,

“checkTime”: “ 20190905111111”,

“forwardTime”: “0”

}

19. 有数据时：

20. {

21. "status": 0, //接口交互成功，0表示成功，其它表示失败

22. "data":

23. [{

24. "ip": “0.0.0.0”， //最新一条数据在最前面，以此类推

25. "uid": "xxxxxxxx"

26. "mac": "1a:2b:3c:4d:5e:6f"，

27. "systemName": "test"，

28. "province": "浙江"，

29. "city": "杭州"，

30. "crop": "1"，

31. "onlineTime": "20190917030000"，

32. "offlineTime": "20190917032959"

33. },

34. {

35. "ip": “0.0.0.0”,

36. "uid": "xxxxxxxx"，

37. "mac": "1a:2b:3c:4d:5e:6f"，

38. "systemName": "test"，

39. "province": "浙江"，

40. "city": "杭州"，

41. "crop": “1”，

42. "onlineTime": "20190917020000"，

43. "offlineTime": "20190917022959" }

44. ]

45. }

查询无数据时返回如下：

46. 无数据时：

47. {

48. "code": 0, //接口交互成功，0表示成功，其它表示失败

49. "data": []

50. }

查询数据首先看data是否为空

如果空，认证失败，返回结果给app

如果非空继续

看data中最新一条数据（list第一条）中的onlineTime是否为空

如果空，认证失败，返回结果给app

如果非空，认证成功，返回结果给app

1. 外网连通性监测 ----逻辑

（数据）

webLoad>5000 || netPingLost=100% App上传数据给APP后台

后台调用云数接口

[healthAnalysis/border/ip/within/get](http://ip:port/healthAnalysis/border/ip/within/get)

[healthAnalysis/border/ip/without/get](http://ip:port/healthAnalysis/border/ip/without/get)

返回结构大概为

{

status：

result：[

ip1{ip:1.1.1.1 xx:xxxx}

]

}

后台将如下数据返回给APP

sendMsg:{

"code": 0, //0表示成功，1 表示失败

"msgtype": 5，//表示携带IP

"moduleName" : “lossPacket”， //模块名称

"testFailStage"：入参中有就传入参中的，没有就传8

"data":{

"ipList": {

"innerIP": [ '10.88.91.5' ],

"outIP": [ '10.88.90.5' ]

}

}

}

App ping innerIP和outIP，并把

"nodePingRes":[

{

"nodeIP":"10.88.91.5", //节点IP

"pingTime":"12.3", //Ping节点耗时，单位：ms

},

{

"nodeIP":"10.88.90.5", //节点IP

"pingTime":"12.3", //Ping节点耗时，单位：ms

},

。。。。。

],

发送个后台

后台根据每个IP的pingTime判断结果发送给app

1. 外网测速检测

（数据）

当"downSpeed"和"uploadSpeed"=0 上传数据给APP后台

后台分三个方面检测外网测速

1. 低选速终端 --- ？

调用云数接口getSlowClient

入参startTime endTime

出参暂时没找到

1. 信道利用率高

调用绿洲接口getStaConnectInfoSpecial

入参

{

shopId: String,

devSN: String,

startTime: Date,

clientMAC: String,

vipFlag:String

}

返回

返回结果, 在data对象中

data:[

{

time: String,

ChB: Number,

TRA: Number,

TRH:Number,

TRL:Number,

RRA: Number,

RRH:Number

RRL:Number

RCC: Number,

SiA: Number,

SiH:Number,

SiL:Number

RPA: Number,

RPH:Number,

RPL:Number,

DA: Number,

DH:Number,

DL:Number,

PDA: Number，

PDH:Number,

PDL:Numeber,

TB：Number

TP：Number

RB：Number

RP：Number

RPB：Number

RBB：Number

RPM：Number

RBM：Number

RPU：Number

RBU：Number

clientScore:Number,

Ap:String,

Rd:Number,

acSN:String,

acName:String

reason:String

},

…

]

对list中的每个chB降序排序，取最大值

如果>60，信道不繁忙

如果<60，信道繁忙

1. 空口广播占比高

调用接口getAPInfoByBSSID（前面有用到过）

入参{"BSSID":"b0f9-63e6-8749", "shopId":"4"}

出参{“apSN”}

调用接口getAsset（接口文档未找到）

入参{typeId，pageNum，pageSize，attributeMap{apSN}}: attributeMap是字符串，将apSN拼接在后面

出参{“uuid”} ： 含义暂不知

同时调用接口绿洲getHistoryStaRiskInfo

参数{

shopId: String,

devSN: String,

startTime: Date,

endTime: Date,

clientMAC: String,

type:Array

}

根据事件类型（type）返回相应字段

data:[

{

time: String,

type: Number,

acSN: String,

acName:String。

authType: Date,

clientIP: String,

apName: String,

radioId: Number,

clientMode: String,

radioType: String,

ssidName: String,

onlieTime: Date,

offlineType: String,

offlineSubType: Number,

clientRxBytes: Number,

clientTxBytes: Number,

clientRxRate: Number,

clientTxRate: Number,

clientScore: Number,

channelBusy: Number,

radioLoad: Number,

rssi: Number,

delay: Number,

drop: Number,

retry: Number,

RUT: Number,

accessFailType: String,

accessFailSubType: Number,

reason: String,

AU1: Number,

AU2": Number,

 AU3: Number,

AU4: Number,

 ARQ: Number,

ARS: Number,

ESTART: Number,

ERQ: Number,

ERS: Number,

EKEY: Number,

ESUC: Number,

EM1: Number,

EM2: Number,

EM3: Number,

 EM4: Number,

DHD: Number,

DHO: Number,

DHR: Number,

DHA: Number,

ARQ: Number,

ARS: Number,

RSA：[

{

RSL：String，

TFR：String

TO：String

TD：String

TR：String

RO：String

}

]

},

…

]

Time降序排序取前三个

调用接口getDeviceConditionByTime

入参StartTime endtime time（time前三个） uuid

出参{

"data": {

"cpuUtilization": 14,

"issueTotal": {

"1": 0,

"2": 0,

"3": 0,

"4": 0,

"5": 0

},

"healthList": [{

"name": "健康度评分",

"value": 9,

"color": 0,

"clickType": null

}, {

"name": "最低分指标",

"value": "信道利用率",

"color": -1,

"clickType": null

}, {

"name": "连接终端数",

"value": 0,

"color": -1,

"clickType": null

}, {

"name": "上行速率",

"value": "0bps",

"color": -1,

"clickType": null

}, {

"name": "下行速率",

"value": "0bps",

"color": -1,

"clickType": null

}],

"software": "",

"ip": "",

"issueList": [],

"type": "ap",

"deviceName": "s2-b318",

"statusList": [{

"name": "在线状态",

"value": "在线",

"color": 0,

"clickType": null

}],

"systemStatusList": [{

"name": "CPU利用率",

"value": "14%",

"color": 0,

"clickType": null

}, {

"name": "内存利用率",

"value": "42%",

"color": 0,

"clickType": null

}],

"memUtilization": 42,

"interval": 300000,

"sn": "219801A0TB9168Q00078",

"radioList": [{

"rtb": 0,

"tx\_channel\_usage": 0,

"tbu": 0,

"rx\_channel\_usage": 0,

"radio\_noise\_floor": -92,

"channel\_usage": 4,

"clientnumber": 0,

"mode": "5G",

"radio\_noise\_floor\_color": 0,

"userCount": 0,

"tbb": 0,

"channel\_usage\_color": 0,

"downBroadcast": 0,

"wlan\_interfere": 0,

"no\_wlan\_interfere": 4,

"radio\_id": 1,

"tbm": 0

}, {

"rtb": 0,

"tx\_channel\_usage": 0,

"tbu": 0,

"rx\_channel\_usage": 0,

"radio\_noise\_floor": -90,

"channel\_usage": 38,

"clientnumber": 0,

"mode": "2.4G",

"radio\_noise\_floor\_color": 0,

"userCount": 0,

"tbb": 0,

"channel\_usage\_color": 0,

"downBroadcast": 0,

"wlan\_interfere": 0,

"no\_wlan\_interfere": 38,

"radio\_id": 2,

"tbm": 0

}]

},

"errorCode": 0,

"message": ""

}

健康度评分是否大于10

>10，空口广播占比不高，结束

否则，最低分指标是否是下行广播占比

是，空口广播占比高

否，占比不高

内网测速：执行命令iperf 一个内网的IP

DHCP慢检测，是后台检测数据字段（app传至后台的）ipDHCP是否大于3000ms：大于则慢，否则不是