SipT 系统配置使用手册

本手册分为 2 部分，第一部分为配置说明，第二部分为使用接口说明。

配置

配置文件为 json 格式，必须为合法的 json 格式文档。所有配置都归属于“global”属性下，如下图

配置分为 3 个对象。

http 配置

http 是系统对外提供服务的控制接口。

port

http 服务监听的端口

listen

http 服务监听的地址，”0.0.0.0”表示在本机所有网络接口上监听。

loglevel

http 服务日志，取值 0-4 分别对应 DEBUG,INFO,WARNING,ERROR,CRITICAL

log 配置

log 指定了系统 log 系统的配置文件

config

log 系统配置文件位置

tag

log 系统主 tag

sip

sip 配置了系统主要配置内容。包括下面 5 个属性

loglevel

sip 系统的 log 等级，取值 5-0，分别对应 trace，debug，info，warning，error，fatal，注意和上面 http 系统的日志系统相区别。

tp

sip 系统的 sip 消息引擎的线程池数量，建议取 CPU 核心数

connected_url

无论 SipT 系统是主叫还是被叫，在每个呼叫接通的时候，post 呼叫数据到指定的 url

call_url

在每个呼叫结束后，post 呼叫数据到指定的 url。

reg_url

在每个注册结束后，post 注册数据到指定的 url。

post_tp

在每个呼叫结束后，post 呼叫数据的时候使用线程池的线程数。

transport

本属性是个数组，数组的每个元素定义个 sip transport，通常我们会为每种（udp，tcp，tls，ws，wss）分别定义一个 transport，但是不会限制，例如定义 2 个 udp transport，那么需要他们定义的端口不一样就行。
transport 对象有如下属性：

type

transport 的类型，取值范围：udp,tcp,tls,ws,wss

listen

transport 监听的地址，建议指定具体网络地址。

port

transport 监听的端口。

cert

加密证书路径，pem 格式，只有 tls 和 wss 端口需要设置，其他类型的 transport 可以不写此属性。

key

加密证书私钥路径，pem 格式，只有 tls 和 wss 端口需要设置，其他类型的 transport 可以不写此属性。

tp

transport 的工作线程池配置的线程数

配置的完整例子如下：

```
{
  "global": {
    "http": {
      "port": 1234,
      "listen": "0.0.0.0",
      "loglevel": 0
    },
    "sip": {
      "loglevel": 5,
      "tp": 5,
      "post_tp": 5,
      "post_url": "http://127.0.0.1:1234/show",
      "transport": [
        {
          "type": "udp",
          "listen": "0.0.0.0",
          "port": 5070,
          "tp": 1
        },
        {
          "type": "udp",
          "listen": "0.0.0.0",
          "port": "5060",
          "tp": 1
        },
        {
          "type": "tcp",
          "listen": "0.0.0.0",
          "port": 5060,
          "tp": 5
        },
        {
          "type": "tls",
          "listen": "192.168.0.177",
          "port": 5061,
          "cert": "config/server.crt",
          "key": "config/server.key",
          "tp": 10
        },
        {
```

```
            "type": "ws",
            "listen": "192.168.0.177",
            "port": 8080,
            "tp": 10
          },
          {
            "type": "wss",
            "listen": "192.168.0.177",
            "port": 8081,
            "cert": "config/server.crt",
            "key": "config/server.key",
            "tp": 10
          }
        ]
      },
      "log": {
        "config": "config/log.ini",
        "tag": "SipT"
      }
    }
}
```

接口说明

下面系统的接口，只给出路径，具体主机和端口自行添加，例如系统接口配置在 http://127.0.0.1:1234,给出的接口路径为/abc,则访问的完整 url 为 http://127.0.0.1:1234/abc

/shutdown

GET，关闭系统，无参数。

/transports

GET，获取系统运行的 transport，无参数，返回 json 数组，如下：

```
[
    "udp:192.168.0.177:5070",
    "udp:192.168.0.177:5060",
    "wss:192.168.0.177:8081",
    "tcp:192.168.0.177:5060",
    "tls:192.168.0.177:5061",
    "ws:192.168.0.177:8080"
]
```

/account/add

POST，添加帐号信息，参数为 json 数组，无返回值。
数组每个元素为一个 json 对象，有如下属性：

user

注册认证的帐号。

pwd

注册认证的密码。

realm

注册认证的域，不设置则客户端任意匹配服务端域，可以不设置此属性，设置后，需要和服务端对应的域一致，本条帐号设置才有效。

scheme

注册认证支持的算法，默认为“digest”，可以不写此属性，为默认值。

所以一条最简单的帐号对象，只需要有 user 和 pwd 两个字段就行了。
注意：所有帐号信息，user 一致的归属为一个帐号信息，在后续注册或者呼叫的时候自动按 user 来配置，有需要的情况可以根据不同域来配置不同的密码，这样可以支持多域 1 个帐号，密码不一致的情况。

```
[
    {
        "user": "1002",
        "pwd": "1002abcde",
        "scheme": "digest",
        "realm": "asterisk"
    },
```

```
    {
        "user": "1002",
        "pwd": "10002"
    },
]
```

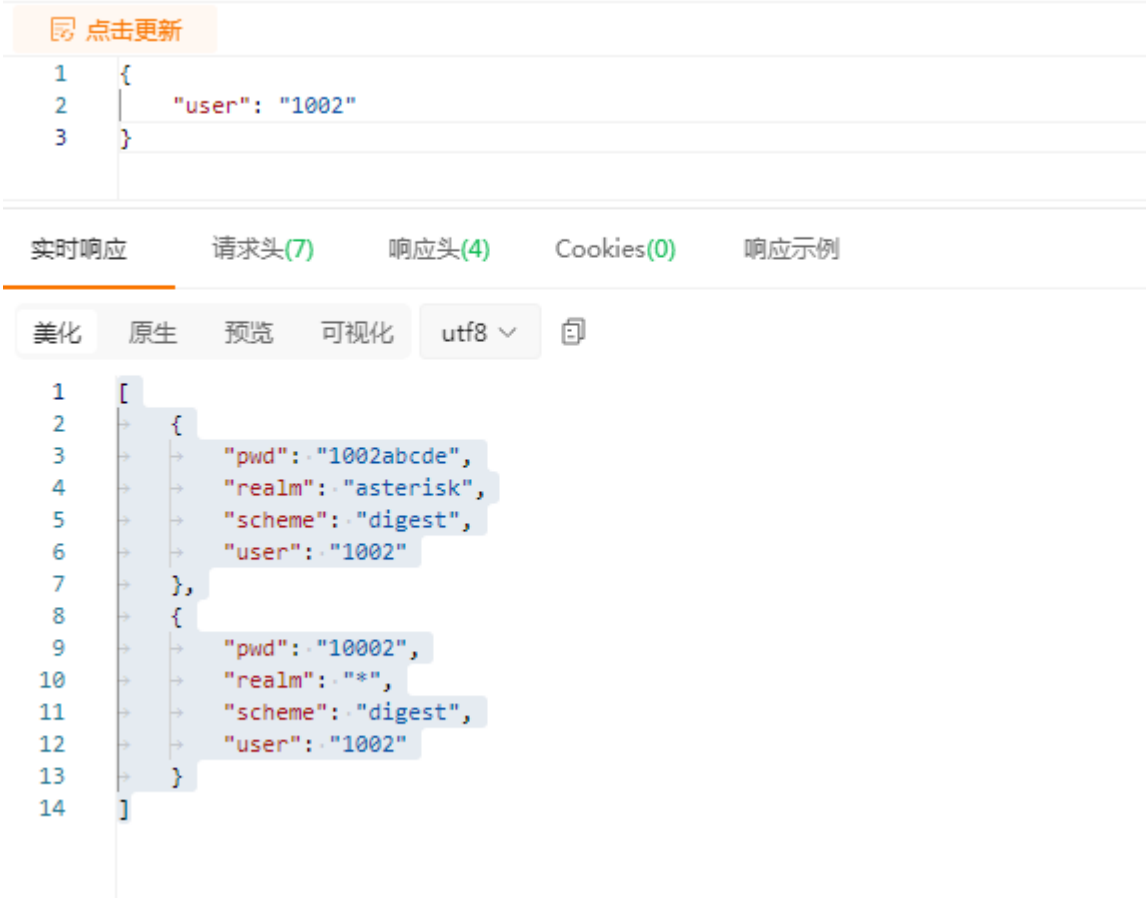
/account/info

POST，获取帐号信息，参数为 json 对象，具体如下：

user

帐号信息。

通过本接口，获取指定帐号下面归属的所有认证信息。返回值为 json 数组，具体如下，上面为请求参数，下面为响应



/account/remove

POST，删除帐号信息，参数的为 json 数组，数组元素为 json 对象，每个对象具有 user 一个字段，可以批量删除帐号下面的认证信息。

user

指定移除的帐号。
例子如下：

```
[
    {
        "user": "1002"
    },
    {
        "user": "1000"
    },
]
```

/register

POST 注册接口，为参数指定的帐号进行注册。参数为 json 数组，数组每个元素为 json 对象，json 对象包含如下属性：

transport

transport 的名字，使用获取 transport 列表中的字符串，用于指定注册信息通过哪个 transport 发送，可以不写此字段，不写则根据其他注册信息自动匹配。

server

SIP 注册服务器，使用标准的 SIP URI，例如：sip:1.2.3.4 sip:1.2.3.4:6060 sip:1.2.3.4:7443;transport=wss
没有指定 transport 的默认都是 udp 协议，没有指定端口，默认就是 5060，如果上一字段 transport 没有指定，则会根据 server 的 transport 来自动匹配走哪个 transport 发送。如果指定上一字段的 transport，会尝试使用指定的 transport 发送。

from

注册消息的 from 字段

reg

注册消息的 to 字段，通常设置和 from 一致

user

注册使用的认证帐号信息，需要提前设置帐号认证信息，否则认证信息将为空。

report

指定呼叫结束时回调的 url，可以不指定，默认为配置文件配置的回调地址，否则提交到指定的 url

expires

注册消息的过期时间，填写整数，单位为秒，可以不设置，不写则默认 60s

hangup

呼入默认动作，1 表示挂机，0 表示接通，可以不设置，默认值为 0。

delay

呼入延迟动作时间，整数，单位为毫秒，可以不设置，默认值为 0，表示无延迟，直接执行默认动作。

cb

呼入回调的 url 地址，可以不设置，默认值为空。cb 为空时，呼入动作由 delay 和 hangup 控制，cb 不为空，则呼入后会使用 POST 请求 cb 指定的 url，post 内容为 json 对象，包含 3 个字段：

from

呼入消息的 from 字段

to

呼入消息的 to 字段

call_id

呼入消息的 call-id 字段

cb 对应的 url 要求返回 json 对象，对象至少包含一个字段

hangup

整数值，1 表示挂机，0 表示接通。

注册示例：

```
[
    {
        "transport": "udp:192.168.2.115:5060",
        "server": "sip:42.51.139.164:5060",
        "from": "sip:1002@42.51.139.164:5060",
        "reg": "1002<sip:1002@42.51.139.164:5060>",
        "user": "1002",
        "expires": 60,
        "hangup":0,
        "delay":3000,
        "cb":"http://127.0.0.1:5000/process_json"
    }
]
```

/UnRegister

POST，注销 SIP 对象。为参数指定的帐号进行注销。参数为 json 数组，数组每个元素为字符串，字符串为；为注册接口的 reg 属性的值。
例如：
[

```
    "1006<sip:1006@client.pbx.aiwcc.com>"
]
```

/MakeCall

POST，批量提交呼叫请求。参数为为 json 数组，数组元素为 json 对象，json 对象包含如下属性：

server

SIP outband 服务器，为 invite 协议的请求行地址，例如：`"server": "sip:*86@20.18.10.165:6065"`，没有指定 transport 的默认都是 udp 协议，没有指定端口，默认就是 5060，如果上一字段 transport 没有指定，则会根据 server 的 transport 来自动匹配走哪个 transport 发送。如果指定上一字段的 transport，会尝试使用指定的 transport 发送。

from

INVITE 消息的 from 字段

to

INVITE 消息的 to 字段

transport

transport 的名字，使用获取 transport 列表中的字符串，用于指定注册信息通过哪个 transport 发送，可以不写此字段，不写则根据其他注册信息自动匹配。

report

指定呼叫结束时回调的 url，可以不指定，默认为配置文件配置的回调地址，否则提交到指定的 url

play

指定媒体文件路径，在接通后，给对端播放 RTP 的内容，文件内容为已经编码的音频，例如目前支持 PCMU 通话，文件内容应当为 PCMU 编码格式。不指定则给对方播放全 0 的数据。

mode

整数，指定媒体文件播放方式，需要结合 play 指定的文件。0 表示不会主动挂断，一直循环播放 play 文件，大于 0，则表示播放次数，例如 1 表示播放 play 文件 1 遍后自动挂机，10 则表示播放 10 遍后挂机。

time

整数，单位秒，控制通话时长。不写本属性，或者设置为 0，则表示不控制，设置为大于 0 的数，表示通话时间长短，例如设置 10，则在接通 10 秒后，本方会主动挂断。

user

帐号认证信息，在呼叫遇到服务器返回鉴权需求的时候，会使用 user 帐号下的认证信息进行认证，不写则没有认证信息。

```
[
    {
        "server": "sip:*86@20.18.10.165:7443;transport=wss",
        "from": "sip:1006@client.pbx.aiwcc.com",
        "to":"sip:*86@client.pbx.aiwcc.com",
        "user": "1006",
        "transport":"wss:192.168.0.123:8081",
        "report":"",
        "play":"",
        "mode":0,
        "time":5
    }
]
```

返回值

返回 json 数组，每个数组元素为 json 对象，包含 3 个属性，分别为

from

对应请求的 from 字段

to

对应请求的 to 字段

call_id

对应请求生成的 call_id，后面和呼叫有关的操作均需要。
如图：

```
[
    {
        "call_id": "3b6beedd1dcc42f3a96c7aad2bd09db5",
        "from": "1000<sip:1000@42.51.139.160>",
        "to": "sip:1234@42.51.139.160"
    }
]
```

/HoldCall

POST, hold 和 unhold 呼叫接口，通过本接口，可以 hold 或者 unhold 通话。参数为 json 数组，数组元素为 json 对象，对象包含 2 个属性：

call_id

发起或者接通呼叫返回的 call_id

hold

整数，0 表示 unhold，1 表示 hold
例子：

```
[
    {
        "call_id": "d06b0685f5bc413d916927eb881344f7",
        "hold":1
    }
]
```

/ReferCall

POST，b-transfer，refer 接口。只能对已经接通的通话有效。参数为 json 数组，数组元素为 json 对象，对象包含 2 个属性：

call_id

发起或者接通呼叫返回的 call_id

to

标准 SIP uri，refer 到的目的地。
例子：

```
[
    {
        "call_id": "91c6e01d869f4282ad91d5b4b466e9d9",
        "to":"sip:1003@client.pbx.aiwcc.com"
    }
]
```

/ReplaceCall

POST，对应 att-transfer 接口。只能对已经接通的通话有效。参数为 json 数组，数组元素为 json 对象，对象包含 2 个属性：

call_id

发起或者接通呼叫返回的 call_id

to

发起或者接通呼叫返回的 call_id，refer-to 字段的目标呼叫
例子：

```
[
    {
        "call_id": "93cd116990da4e3785827656c7f54b3a",
        "to": "f1e3a1564b454d5e816ee92c68a6fde0"
    }
]
```

/hangup

POST，挂机接口，通过本接口可以挂断指定的通话。参数为 json 数组，数组元素为 json 对象，json 对象只包含一个属性

call_id

在呼叫接口中，指定的返回的 call_id 字段

请求示例：

```
[
    {
        "call_id": "sip:1000@client.pbx.aiwcc.com"
    }
]
```

/incoming/config

POST，配置呼入动作接口，通过本接口可以配置指定的注册账号的呼入动作，如果指定的配置已经存在则覆盖更新当前值。参数为 json 数组，数组元素为 json 对象，json 对象只包含 4 个属性（属性同注册接口中含义取值一致，参考/register）

reg

hangup

delay

cb

如下示例：

```
[
    {
        "reg": "1002<sip:1002@42.51.139.164:5060>",
        "hangup":0,
        "delay":3000,
        "cb":""
    }
]
```

回调接口

注册回调

回调通过 post 一个 json 对象告诉注册请求的结果，包含 3 个字段

code

SIP 注册消息的响应代码。

From

SIP 消息的 from 字段

reason

SIP 消息注册的 reason

注册认证失败的结果如下：


```
{
	"code": 401,
	"from": "sip:1006@client.pbx.aiwcc.com",
	"reason": "Unauthorized"
}
```


注册成功例子：


```
{
	"code": 200,
	"from": "sip:1006@client.pbx.aiwcc.com",
	"reason": "OK"
}
```


呼叫回调

当呼叫结束时，给指定 url 通过 POST 请求，发送一个 json 对象

total_duration：

呼叫持续时间，单位毫秒

cause

呼叫消息的最终响应代码

cause_text

呼叫消息最终响应

connect_delay

呼叫开始到接通的时间，单位毫秒

connect_duration

呼叫通话持续的时间，单位毫秒

ringing

呼叫收到 180 或者 183 信令的时间戳，13 位长，单位为毫秒，如果整个呼叫过程没有收到 180 或者 183，值为空字符串“”

connected

呼叫接通的时间戳，13 位长，单位为毫秒，如果整个呼叫过程未接通，值为空字符串“”

disconnected

呼叫结束的时间戳，13 位长，单位为毫秒

from

sip 消息的 from 字段，包括 tag

to

sip 消息的 to 字段，如果接通会有相应的 tag

audio

呼叫中的 rtp 流信息，如下：

```
"audio": [{
        "clock": 8000,
        "codec": "PCMU",
        "pt": 0,
```

```
            "rx": {
                "avg_bps": 59879,
                "discard": 1,
                "dup": 0,
                "loss": 21,
                "reorder": 0,
                "total": 428
            },
            "srtp": "Not active",
            "tx": {
                "avg_bps": 63937,
                "discard": 0,
                "dup": 0,
                "loss": 0,
                "reorder": 0,
                "total": 457
            }
        }]
```


clock：语音流时钟

codec：编码
pt：编码的 rtp payload 字段
rx.avg_bps:平均码率
rx.discard:丢弃包数量
rx.dup：重复包数量
rx.loss：丢失包数量
rx.reorder：次序错误的包数量
total：总的包数量。
tx 表示发送端的数据，含义同上 rx。

失败呼叫：


```
{
    "audio": [{
            "clock": 8000,
            "codec": "PCMU",
            "pt": 0,
            "rx": {
                "avg_bps": 0,
                "discard": 0,
                "dup": 0,
```

```
                "loss": 0,
                "reorder": 0,
                "total": 0
            },
            "srtp": "Not active",
            "tx": {
                "avg_bps": 62822,
                "discard": 0,
                "dup": 0,
                "loss": 0,
                "reorder": 0,
                "total": 8
            }
        }],
        "call_id": "5e8e5682b1db4f57916c756e8f45eb52",
        "cause": 480,
        "cause_text": "Temporarily Unavailable",
        "connect_delay": 0,
        "connect_duration": 0,
        "ringing":"",
        "connected":"",
        "disconnected":"1694764203334",
        "from": "sip:1006@client.pbx.aiwcc.com",
        "to": "sip:1007@client.pbx.aiwcc.com",
        "total_duration": 509
}
```


成功呼叫：


```
 {
        "audio": [{
            "clock": 8000,
            "codec": "PCMU",
            "pt": 0,
            "rx": {
                "avg_bps": 59879,
                "discard": 1,
                "dup": 0,
                "loss": 21,
                "reorder": 0,
                "total": 428
            },
            "srtp": "Not active",
            "tx": {
```

```
                "avg_bps": 63937,
                "discard": 0,
                "dup": 0,
                "loss": 0,
                "reorder": 0,
                "total": 457
            }
        }],
        "call_id": "e2aa1447f8dc496f97060d238d61fc1e",
        "cause": 200,
        "cause_text": "OK",
        "connect_delay": 367,
        "connect_duration": 9135,
        "ringing":"",
        "connected":"1694764194562",
        "disconnected":"1694764203334"
        "from": "sip:1006@client.pbx.aiwcc.com",
        "to": "sip:*86@client.pbx.aiwcc.com",
        "total_duration": 9502
}
```


接通回调

from

呼叫的 from 字段

to

呼叫的 to 字段

call_id

呼叫的 call_id

ringing

呼叫收到 180 或者 183 信令的时间戳，13 位长，单位为毫秒，如果整个呼叫过程没有收到 180 或者 183，值为空字符串“”

connected

呼叫接通的时间戳，13 位长，单位为毫秒，如果整个呼叫过程未接通，值为空字符串“”

呼叫失败无回调

呼叫接通回调例子


```
{
    "call_id": "e2aa1447f8dc496f97060d238d61fc1e",
    "from": "sip:1006@client.pbx.aiwcc.com",
    "to": sip:*86@client.pbx.aiwcc.com，
    "connected":"1694764194562"，
    "ringing":"",
}
```