WebAPI Specification

REST Interface Specification for the WebSBC™

Version 1.1 May, 2013

sansay™

TABLE OF CONTENTS

2 Introduction

2 Authentication And Security

3 API Example Message Flow – RTC to RTC Relay

4 API Example Message Flow – RTC to SIP

5 REST Resources

5 Example Message Formats

5 Status Polling

6 Media Relay Allocation

8 Media Relay Updates

9 Media Relay Refresh

10 Media Relay De-Allocation

11 SIP Outbound

13 SIP Outbound – Reverse REST call Answer

14 SIP Session Refresh

14 SIP Session Release

15 SIP Register

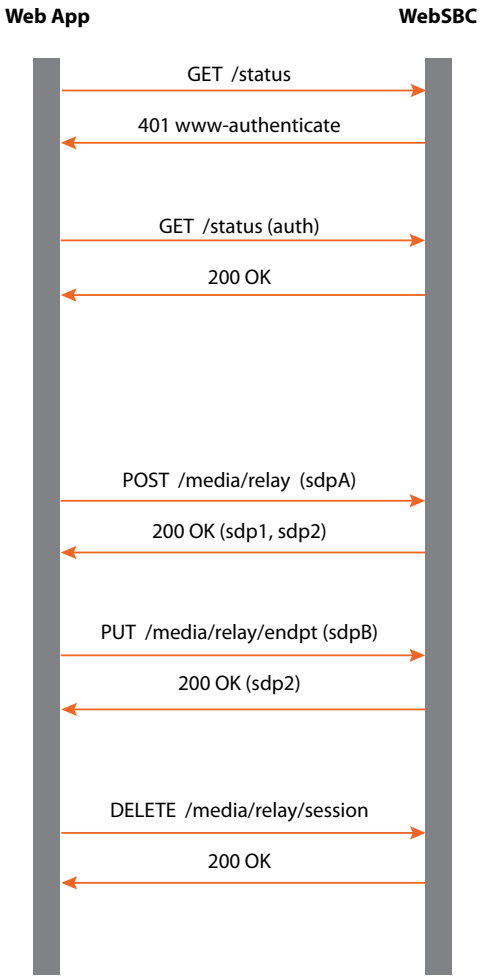
INTRODUCTION

This document is the RAPID API specification for controlling the Sansay WebSBC™. The Sansay RAPID API uses a REST based HTTP interface for the purpose of providing media plane services to a website application. The RAPID API also provides SIP connectivity functions allowing for a website application to interact directly with SIP based networks. In order to address the bidirectional nature of SIP messaging, an additional REST interface is required for the WebSBC to Application Server direction. The HTTP message bodies used by this REST API are JSON formatted.

AUTHENTICATION AND SECURITY

HTTP basic authentication is used by the WebSBC in order to grant access to the service. For that reason, HTTPS must always be used by the website application in order to hide the credential information when sending across the open internet. The WebSBC and Application Server will each hold signed certificates and will be provisioned with the authentication credentials and the root CA certificate.

API EXAMPLE MESSAGE FLOW – RTC TO RTC RELAY

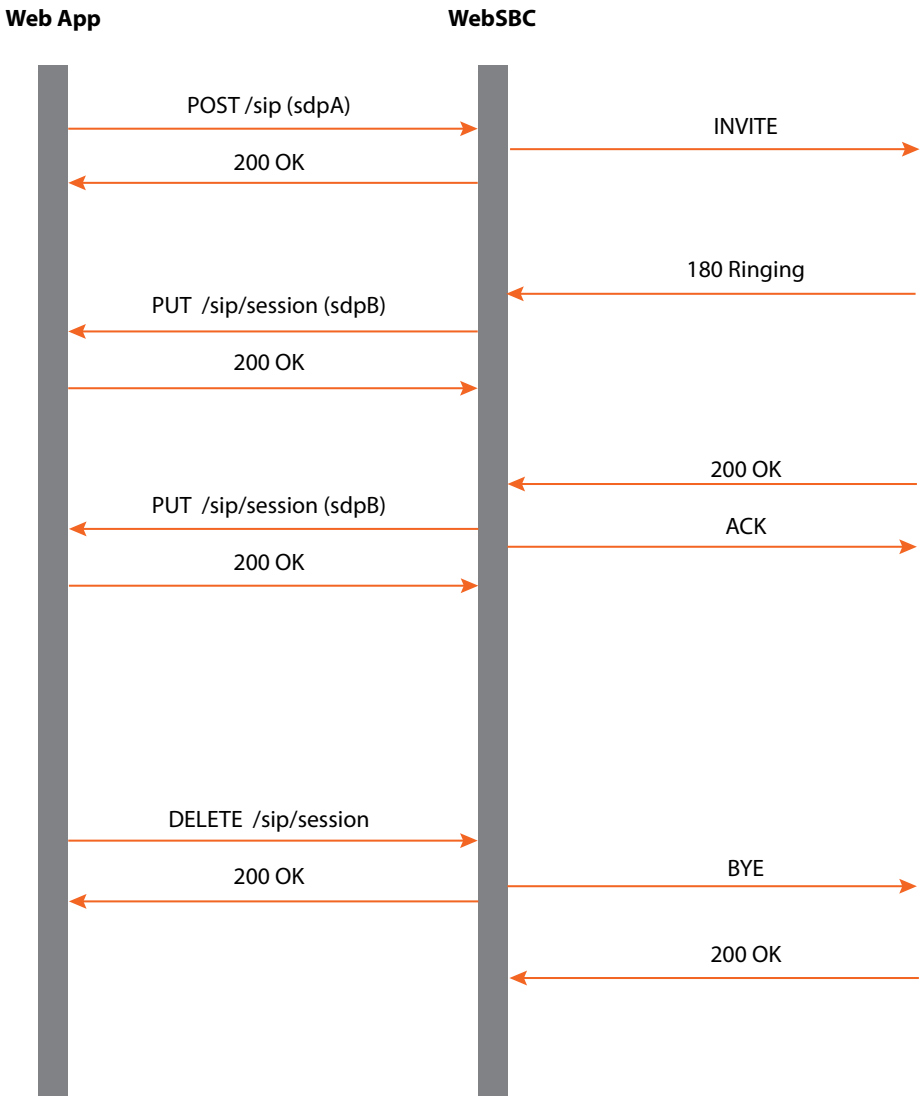


The above message flow begins with the website application server creating an HTTPS session with the WebSBC. The 401 challenge will occur if the Authentication header is missing from the request. It is recommended that the website application always include the Authentication header in all requests. The WebSBC may choose to not send a 401 challenge to prevent it from being detected by attack scanners.

The web app in the example above, begins by sending a GET /status request. The WebSBC will respond with statistics pertinent to that web application instance. The GET /status request can also be used as a periodic poll to the WebSBC so that the web application can continuously know the state of its readiness.

The POST requests in the example are used by the web application to create a set of media relay ports. The web app sends the SDP A of the originating client and the WebSBC responds with the two relay ports contained within the SDP 1 and SDP 2 respectively. The web app then updates the second relay port with the SDP B information gathered from the termination client. The web app then finally terminates the relay session by sending a DELETE request to the session url.

API EXAMPLE MESSAGE FLOW – RTC TO SIP



REST RESOURCES

The following is a list of the REST based resources offered by the WebSBC:

/status - URL for retrieving system status.

/media/relay - URL for allocating relay sessions.
/media/relay/[session-id] - URL for manipulating existing relay sessions.
/media/relay/[session-id]/[endpoint-id] - URL for manipulating endpoint configurations within a relay session.

/sip - URL for creating outbound SIP sessions.
/sip/[session-id] - URL for manipulating existing SIP sessions.
/sip/[session-id]/[endpoint-id] - URL for manipulating endpoint configurations within a SIP session.
/sip/register - URL for creating SIP subscribers into a SIP network.

EXAMPLE MESSAGE FORMATS

STATUS POLLING

The website server can create and keep HTTPS sessions alive by periodically sending a GET request for status.

```
GET /status HTTP/1.1
Host: sansay.websbc.west
Authorization: Basic account-name-and-password
Accept: application/json
X-Sansay-API-Version: 1.1
[blank line]
```

If the account name and password are valid, the following response will be returned:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nn
[blank line]
{
   “state” : “online”,
   “active_relays” : 1284,
   “allowed_relays” : 2000
}
```

If the website server is using the SIP portion of the API then it can tell the webSBC where it wants to receive messages for the reverse direction REST interface.

```
GET /status HTTP/1.1
Host: sansay.websbc.west
Authorization: Basic account-name-and-password
Accept: application/json
Content-Type: application/json
Content-Length: nnn
X-Sansay-API-Version: 1.1
[blank line]
{
   "rest_address" : "192.168.0.12:443",
   "credentials" : "reverse-account-name-and-password"
}
```

If the account name and password are valid, the following response will be returned:

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nn
[blank line]
{
   "state" : "online",
   "active_relays" : 1284,
   "allowed_relays" : 2000
}
```

MEDIA RELAY ALLOCATION

The website server can allocate generic media relay ports through the base /media/relay url. The request is designed to re-use many of the ROAP variable names and concepts in order to make it simpler for web applications already familiar with WebRTC and ROAP. For instance, the website will create a unique "offererSessionID" during the allocation and the WebSBC will respond with the "answererSessionID". That way both applications can control their own session ID assignment in order to optimize performance. The same technique is also applied to endpoint IDs.

```
POST /media/relay HTTP/1.1
Host: sansay.websbc.west
Authorization: Basic account-name-and-password
Accept: application/json
Content-Type: application/json
Content-Length: nnn
X-Sansay-API-Version: 1.1
[blank line]
{
   “offererSessionId”: “12324ABCD”,
   “seq”: 1,
   “services”:
   {
      “qos”:“enable”,
      “trace”:“enable”,
      “transcode”:“none”,
      “record”:“none”,
      “filter”:“audio_only”
   },
   “session_descriptions”:
  [
        {
          “offererEndpointId”: “12324ABCD/1”,
          “nat”:“enable”,
          “sdp”:“v=0\r\no=- 2890844526 2890842807 IN IP4 192.0.2.1\r\ns=\r\nc=IN IP4
192.0.2.1\r\nt=2873397496 2873404696\r\nm=audio 49170 RTP/AVP 0\r\n”
          },
          {
          “offererEndpointId”: “12324ABCD/2”,
          “nat”:“disable”,
          “sdp”:“none”
          }
   ]
}
```

```
HTTP/1.1  201 Created
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
   “offererSessionId” :  “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 1,
   “expires” :  600,
   “session_descriptions” :
   [
        {
         “offererEndpointId” :  “12324ABCD/1”,
         “answererEndpointId” : “398448ABCDEF/10734”,
         “nat” : “enable”,
         “sdp” : “v=0\r\no=sansay 188 1 IN IP4 192.188.2.3\r\ns= webSBC\r\nc=IN IP4 192.188.2.3\r\nt=0
0\r\nm=audio 10734 RTP/AVP 0\r\n”
},
{
      “offererEndpointId” :  “12324ABCD/2”,
      “answererEndpointId” : “398448ABCDEF/10736”,
      “nat” : “disable”,
      “sdp” : “v=0\r\no=sansay 188 2 IN IP4 192.188.2.3\r\ns= webSBC\r\nc=IN IP4 192.188.2.3\r\nt=0
0\r\nm=audio 10736 RTP/AVP 0\r\n”
             }
   ]
}
```

MEDIA RELAY UPDATES

The website server can update the relay session in order to provide a missing SDP or to modify an SDP using the end-point id url.

```
PUT  /media/relay/398448ABCDEF/10736  HTTP/1.1
Host:  www.sansay.websbc.west
Authorization:  Basic  account-name-and-password
Accept:  application/json
Content-Type:  application/json
Content-Length:  nnn
```

```
X-Sansay-API-Version:  1.1
[blank line]
{
   “offererSessionId”:  “12324ABCD”,
   “answererSessionId”:“398448ABCDEF”,
   “seq”:2,
   “session_description”:
   {
      “offererEndpointId”:  “12324ABCD/2”,
      “answererEndpointId”:“398448ABCDEF/10736”,
      “nat”:“disable”,
      “sdp”:“v=0\r\no=- 2890844526 2890842807 IN IP4 192.0.2.5\r\ns=\r\nc=IN IP4 192.0.2.1\r\nt=2873397496
2873404696\r\nm=audio 49178 RTP/AVP 0\r\n”
   }
}
```

```
HTTP/1.1  201 Created
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
   “offererSessionId”:  “12324ABCD”,
   “answererSessionId”:“398448ABCDEF”,
   “seq”:2,
   “session_description”:
   {
     “offererEndpointId”:  “12324ABCD/2”,
      “answererEndpointId”:“398448ABCDEF/10736”,
      “nat”:“disable”,
      “sdp”:“v=0\r\no=sansay 188 1 IN IP4 192.188.2.3\r\ns= webSBC\r\nc=IN IP4 192.188.2.3\r\nt=0
0\r\nm=audio 10736 RTP/AVP 0\r\n”
   }
}
```

MEDIA RELAY REFRESH

The website server can extend the relay allocation expiration by sending a session refresh request to the session url.

```
PUT /media/relay/398448ABCDEF HTTP/1.1
Host: sansay.websbc.west
Authorization: Basic account-name-and-password
Accept: application/json
Content-Type: application/json
Content-Length: nn
X-Sansay-API-Version: 1.1
[blank line]
{
   "offererSessionId" : "12324ABCD",
   "answererSessionId" : "398448ABCDEF",
   "seq" : 3
}
```

```
HTTP/1.1 201 Created
Content-Type: application/json
Content-Length: nnn
[blank line]
{
   "offererSessionId" : "12324ABCD",
   "answererSessionId" : "398448ABCDEF",
   "seq" : 3,
   "expires" : 600
}
```

MEDIA RELAY DE-ALLOCATION

The website server can remove a relay allocation by sending a session delete request to the session url.

```
DELETE /media/relay/398448ABCDEF HTTP/1.1
Host: sansay.websbc.west
Authorization: Basic account-name-and-password
Accept: application/json
Content-Type: application/json
Content-Length: nn
X-Sansay-API-Version: 1.1
[blank line]
```

```
{
   “offererSessionId” : “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 4
   “reason” : “release”
}
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: nnn
[blank line]
{
   “offererSessionId” : “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 4,
   “expires” : 0
}
```

SIP OUTBOUND

The website server can instruct the WebSBC to place an outbound SIP call.

```
POST /sip HTTP/1.1
Host: www.sansay.websbc.west
Authorization: Basic account-name-and-password
Accept: application/json
Content-Type: application/json
Content-Length: nnn
X-Sansay-API-Version: 1.1
[blank line]
{
   “offererSessionId” : “12324ABCD”,
```

```
seq": 1,
   "contact": "192.188.0.23:443"
   "credentials": "reverse-account-name-and-password",
   "services":
   {
      "qos": "enable",
      "trace": "enable",
      "transcode": "none",
      "record": "none",
      "filter": "audio_only"
   },
   "route":
   {
      "ani_clid": "Martin"
      "ani": "8581230987"
      "dnis_clid": "Max"
      "dnis": "5124858821"
      "orig_trunk": "102"
      "dest_trunk": "49995"
   },
   "session_description":
   {
      "offererEndpointId": "12324ABCD/1",
      "nat": "enable",
      "sdp": "v=0\r\no=- 2890844526 2890842807 IN IP4 192.0.2.1\r\ns=\r\nc=IN IP4 192.0.2.1\r\nt=2873397496
2873404696\r\nm=audio 49170 RTP/AVP 0\r\n"
   }
}
```


```
HTTP/1.1  201 Created
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
   "offererSessionId": "12324ABCD",
   "answererSessionId": "398448ABCDEF",
   "seq": 1,
   "expires": 600,
```

SIP OUTBOUND – REVERSE REST CALL ANSWER

The WebSBC will use the opposite REST interface within the App Server in order to send ringing and answer indications.

```
PUT  /sip/12324ABCD  HTTP/1.1
Host:  www.AppServer1.com
Authorization:  Basic  account-name-and-password
Accept:  application/json
Content-Type:  application/json
Content-Length:  nnn
X-Sansay-API-Version:  1.1
[blank line]
{
   “offererSessionId”:  “12324ABCD”,
   “answererSessionId”:“398448ABCDEF”,
   “seq”:2,
   “event”:“answer”
   “session_description”:
   {
      “offererEndpointId”:  “12324ABCD/1”,
      “answererEndpointId”:“398448ABCDEF/49170”,
      “nat”:“enable”,
      “sdp”:“v=0\r\no=- 2890844526 2890842807 IN IP4 192.0.2.1\r\ns=\r\nc=IN IP4 192.0.2.1\r\nt=2873397496
2873404696\r\nm=audio 49170 RTP/AVP 0\r\n”
   }
}
```

```
HTTP/1.1  201 Created
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
   “offererSessionId”:  “12324ABCD”,
   “answererSessionId”:“398448ABCDEF”,
   “seq”:2,
}
```

SIP SESSION REFRESH

The website server can extend the SIP expiration by sending a session refresh request to the session url.

```
PUT  /sip/398448ABCDEF  HTTP/1.1
Host:  sansay.websbc.west
Authorization:  Basic  account-name-and-password
Accept:  application/json
Content-Type:  application/json
Content-Length:  nn
X-Sansay-API-Version:  1.1
[blank line]
{
   “offererSessionId” :  “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 3
}
```

```
HTTP/1.1  201 Created
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
   “offererSessionId” :  “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 3,
   “expires” :  600
```

SIP SESSION RELEASE

The website server can release a SIP session by sending a session delete request to the session url.

```
DELETE  /sip/398448ABCDEF  HTTP/1.1
Host:  sansay.websbc.west
Authorization:  Basic  account-name-and-password
Accept:  application/json
Content-Type:  application/json
Content-Length:  nn
```

```
X-Sansay-API-Version:  1.1
[blank line]
{
   “offererSessionId” :  “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 4
   “reason” : “release”
}
```

```
HTTP/1.1  200  OK
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
   “offererSessionId” :  “12324ABCD”,
   “answererSessionId” : “398448ABCDEF”,
   “seq” : 4,
   “expires” :  0
}
```

SIP REGISTER

The website server can present WebRTC clients as SIP registered clients to a SIP network.

```
POST  /sip/register  HTTP/1.1
Host:  sansay.websbc.west
Authorization:  Basic  account-name-and-password
Accept:  application/json
Content-Type:  application/json
Content-Length:  nn
X-Sansay-API-Version:  1.1
[blank line]
{
   “offererRegisterId” :  “12324ABCD”,
   “seq” : 3
   “username” : “john.doe”
```

```
    “domain” : “sansay.rocks”
    “password” : “888”
}


HTTP/1.1  201 Created
Content-Type:  application/json
Content-Length:  nnn
[blank line]
{
    “offererRegisterId” : “12324ABCD”,
    “answererRegisterId” : “398448ABCDEF”,
    “seq” : 3,
    “expires” : 600
}
```

Corporate Headquarters:
Sansay, Inc.
4350 La Jolla Village Dr. Ste. 888
San Diego, CA 92122

Toll Free: +1 888.889.8906
Office: +1 858.754.2200
Fax: +1 858.550.2044

www.sansay.com