Stream capturing from a SIP call

Overview

WCS can work as a WebRTC-SIP gateway. In this case, audio and video stream of a SIP call made through WCS can be captured and played in a browser or republished to another server.

Typical use case

1. A video call is established between WCS and a SIP device (SIP MCU, conference server or a SIP softphone)
2. WCS receives audio and video data from this SIP device
3. The WCS server redirects the received audio and video traffic to an RTMP server or another device capable of receiving and processing an RTMP stream

Supported protocols

- RTMP
- SIP

Supported SIP codecs

- Video: H.264, VP8
- Audio: G.711, Speex

Supported RTMP codecs

- Video: H.264
- Audio: AAC, G.711, Speex

Capturing and republishing of SIP calls is managed using REST API queries.

REST-queries

General rules

1. Each SIP call can be associated with just one RTMP stream. If a new SIP call is initiated with the same RTMP URL and stream name (rtmpUrl+rtmpStream) as the existing call, that second call is declined by the server with the HTTP status of 409 Conflict. However, publishing of a call to an RTMP stream using the /push/startup REST query does not limit the number of RTMP streams created for one call.
2. SIP Call ID of a call must be unique. An attempt to initiate a new SIP call with an already existing Call ID is declined by the WCS server with the HTTP status of 409 Conflict.

The REST query is an HTTP/HTTPS POST query as follows:
- HTTP:http://sip-as-rtmp.flashphoner.com:8081/rest-api/call/startup
- HTTPS:https://sip-as-rtmp.flashphoner.com:8444/rest-api/call/startup

Where:
- test.flashphoner.com - is the address of the WCS server
- 8081 - is the standard REST / HTTP port of the WCS server
- 8444 - is the standard HTTPS port
- rest-api - is the required part of the URL
- /call/startup - the REST method used

REST-methods and response statuses

REST-method	Example of REST-query	Example of REST-response body	Response status

/call/startup	{
	"callId":"123456711",
	"callee":"10000",
	"toStream":"stream1",
	"rtmpUrl":"rtmp://localhost:1935/live/",
	"rtmpStream":"rtmp_stream1",
	"hasAudio":"true",
	"hasVideo":"true",
	"sipLogin":"10009",
	"sipAuthenticationName":"10009",
	"sipPassword":"1234",
	"sipDomain":"226.226.225.226",
	"sipOutboundProxy":"226.226.225.226",
	"sipPort":"5060",
	"appKey":"defaultApp",
	"sipRegisterRequired":"false"
	}
		{}	200 - The call is accepted for processing
			
			409 - Conflict with an existing RTMP URL

/call/find	{"status" : "ESTABLISHED"} or {"callId":"R08NQya-5NMe5v7q-JNkboaS-CGMlFi"}	[{ "custom": {}, "nodeId": null, "appKey": null, "sessionId": null, "callId": "R08NQya-5NMe5v7q-JNkboaS-CGMlFi", "parentCallId": null, "incoming": false, "status": "ESTABLISHED", "sipStatus": 200, "rtmpUrl": null, "rtmpStream": null, "streamName": null, "rtmpStreamStatus": null, "caller": "001", "callee": "002", "hasAudio": true, "hasVideo": false, "sdp": null, "visibleName": "001", "inviteParameters": null, "mediaProvider": "Flash", "sipMessageRaw": null, "isMsrp": false, "target": null, "holdForTransfer": false }]	200 - call is found 404 - call with the given parameters is not found

/call/find_all	{}	<pre>[{
		"custom": {},
		"nodeId": null,
		"appKey": null,
		"sessionId": null,
		"callId": "R08NQya-5NMe5v7q-
		JNkboaS-CGMlFi",
		"parentCallId": null,
		"incoming": false,
		"status": "ESTABLISHED",
		"sipStatus": 200,
		"rtmpUrl": null,
		"rtmpStream": null,
		"streamName": null,
		"rtmpStreamStatus": null,
		"caller": "001",
		"callee": "002",
		"hasAudio": true,
		"hasVideo": false,
		"sdp": null,
		"visibleName": "001",
		"inviteParameters": null,
		"mediaProvider": "Flash",
		"sipMessageRaw": null,
		"isMsrp": false,
		"target": null,
		"holdForTransfer": false
		}]</pre>
			200 - calls are found
			404 - calls are not found

/call
/terminate
	<pre>{"callId" : "becee2c0-13b4-11e7-b817-
	c1649197cae8"}</pre>
			200 - call is terminated
			404 - call is not found

/call
/send_dtmf
	<pre>{
	"callId" : "52173e00-13b6-11e7-b817-
	c1649197cae8",
	"dtmf":"9",
	"type":"RFC2833"
	}</pre>
			200 - DTMF is sent
			404 - call is not found

/call
/inject_stream
	<pre>{
	"callId": "ad5ac8a0-b518-11e7-82c7-
	999b45e427ba",
	"streamName": "mixer1_stream"
	}</pre>
			200 - audio stream is added to the call
			404 - call is not found

Parameters

Parameter name	Description	Example
callId	SIP Call ID - a unique identifier string	Xq2tlLcX89tTjaji
callee	SIP callee	10001
toStream	Name of the stream on the WCS server the call is published to	call_stream1

rtmpUrl	RTMP URL - address of the RTMP server	rtmp://rtmp-server.flashphoner.com:1935/live Here live - is the name of the RTMP application. Also, in RTMP URL the instance name and the query string can be specified, for example: rtmp://rtmp-server.flashphoner.com:1935/live/_definst_?param1=value1&param2=value2
rtmpStream	Name of the RTMP stream on the RTMP server	streamName2
hasAudio	If true, SDP will have the 'sendrecv' parameter in audio. If false, it gets 'recvonly'.	true
hasVideo	If true, SDP will have the 'sendrecv' parameter in video. If false, it gets 'recvonly'.	true
status	Call status on the WCS server	ESTABLISHED The complete list of statuses is available in the Call Flow (see the CallStatusEvent method).
sipStatus	Associated SIP-status	200
rtmpStreamStatus	Status of the RTMP stream	RTMP_STREAM_ACTIVE RTMP_STREAM_WAIT - RTMP-stream is initializing RTMP_STREAM_ACTIVE - RTMP-stream has initialized and connection is established RTMP_CONNECTION_LOST - RTMP-connection is lost RTMP_CONNECTION_FAILED - RTMP-connection was not established
caller	SIP caller	
visibleName	Displayed name of the caller	
mediaProvider	NOT USED	NOT USED

SDP parameters recvonly and sendrecv

There are two main modes for SIP-REST calls:

1. sendrecv

```
v=0
o=Flashphoner 0 1437391553771 IN IP4 sip.flashphoner.com
s=Flashphoner/1.0
c=IN IP4 sip.flashphoner.com
t=0 0
m=audio 31022 RTP/AVP 8 0
c=IN IP4 46.101.139.106
a=rtpmap:8 pcma/8000
a=rtpmap:0 pcmu/8000
a=ptime:20
a=rtcp:31023 IN IP4 sip.flashphoner.com
a=sendrecv
a=ssrc:1478013757 cname:rtp/audio/Xq2tlLcX89tTjaji
m=video 31024 RTP/AVP 112 113
c=IN IP4 sip.flashphoner.com
a=rtpmap:112 H264/90000
a=fmtp:112 packetization-mode=1; profile-level-id=420020
a=rtpmap:113 H264/90000
a=fmtp:113 packetization-mode=0; profile-level-id=420020
a=rtcp-fb:* ccm fir
a=rtcp-fb:* nack
a=rtcp-fb:* nack pli
a=rtcp:31025 IN IP4 sip.flashphoner.com
a=sendrecv
a=ssrc:979076678 cname:rtp/video/Xq2tlLcX89tTjaji
```

2. recvonly

```
hasAudio: false
hasVideo: false

v=0
o=Flashphoner 0 1437391553771 IN IP4 sip.flashphoner.com
s=Flashphoner/1.0
c=IN IP4 sip.flashphoner.com
t=0 0
m=audio 31022 RTP/AVP 8 0
c=IN IP4 46.101.139.106
a=rtpmap:8 pcma/8000
a=rtpmap:0 pcmu/8000
a=ptime:20
a=rtcp:31023 IN IP4 sip.flashphoner.com
a=recvonly
a=ssrc:1478013757 cname:rtp/audio/Xq2tlLcX89tTjaji
m=video 31024 RTP/AVP 112 113
c=IN IP4 sip.flashphoner.com
a=rtpmap:112 H264/90000
a=fmtp:112 packetization-mode=1; profile-level-id=420020
a=rtpmap:113 H264/90000
a=fmtp:113 packetization-mode=0; profile-level-id=420020
a=rtcp-fb:* ccm fir
a=rtcp-fb:* nack
a=rtcp-fb:* nack pli
a=rtcp:31025 IN IP4 sip.flashphoner.com
a=recvonly
a=ssrc:979076678 cname:rtp/video/Xq2tlLcX89tTjaji
```

In both cases WCS does not send RTP audio and video traffic, because it is the REST client that is the initiator of the call, which is not the source of audio and video streams. WCS can explicitly set in SDP that there will be no audio and video traffic from its side (the 'recvonly' mode).

If your SIP-device is a softphone or another SIP phone, most likely it will drop calls (in the 'sendrecv' mode) within approximately a minute after connection is established. This is because of lack of RTP traffic from WCS.

Some softphones correctly support the 'recvonly' mode, for example, MicroSIP.
In other softphones like Bria, RTP activity timer can be manually increased to provide longer duration of a call in the 'sendrecv' mode.

If your SIP device is an MCU or a SIP conference server, it should work correctly with the 'recvonly' mode, and long calls can be established.

Additional status information

WCS uses the built-in 'callApp' application to send intermediate statuses.

Examples

- TRYING, RTMP_STREAM_WAIT

```
{
  "nodeId" : "w9NiNKZCtjK6C4vz1zVnzGWBJGkA2Cke@192.168.88.101",
  "appKey" : "callApp",
  "sessionId" : "127.0.0.1:1403649870519623722",
  "callId" : "Xq2tlLcX89tTjaji_3",
  "incoming" : false,
  "status" : "TRYING",
  "sipStatus" : 100,
  "rtmpUrl" : "rtmp://rtmp.flashphoner.com:1935/live",
  "rtmpStream" : "streamName2",
  "rtmpStreamStatus" : "RTMP_STREAM_WAIT",
  "caller" : "3000",
  "callee" : "3002",
  "hasAudio" : true,
  "hasVideo" : true,
  "visibleName" : "3000",
  "mediaProvider" : "Flash",
  "isMsrp" : false
}
```


- ESTABLISHED, RTMP_STREAM_ACTIVE


```
{
  "nodeId" : "w9NiNKZCtjK6C4vz1zVnzGWBJGkA2Cke@192.168.88.101",
  "appKey" : "callApp",
  "sessionId" : "127.0.0.1:1403649870519623722",
  "callId" : "Xq2tlLcX89tTjaji_3",
  "incoming" : false,
  "status" : "ESTABLISHED",
  "sipStatus" : 200,
  "rtmpUrl" : "rtmp.flashphoner.com:1935/live",
  "rtmpStream" : "streamName2",
  "rtmpStreamStatus" : "RTMP_STREAM_ACTIVE",
  "caller" : "3000",
  "callee" : "3002",
  "hasAudio" : true,
  "hasVideo" : true,
  "visibleName" : "3000",
  "mediaProvider" : "Flash",
  "isMsrp" : false
}
```


- ESTABLISHED, RTMP_CONNECTION_LOST

```
{
  "nodeId" : "w9NiNKZCtjK6C4vz1zVnzGWBJGkA2Cke@192.168.88.101",
  "appKey" : "callApp",
  "sessionId" : "127.0.0.1:1403649870519623722",
  "callId" : "Xq2tlLcX89tTjaji_3",
  "incoming" : false,
  "status" : "ESTABLISHED",
  "sipStatus" : 200,
  "rtmpUrl" : "rtmp.flashphoner.com:1935/live",
  "rtmpStream" : "streamName2",
  "rtmpStreamStatus" : "RTMP_CONNECTION_LOST",
  "caller" : "3000",
  "callee" : "3002",
  "hasAudio" : true,
  "hasVideo" : true,
  "visibleName" : "3000",
  "mediaProvider" : "Flash",
  "isMsrp" : false
}
```

These are notifications that are sent only locally on the server side via the internal REST interface. See theApplication managementsection to get more information about internal REST applicaitons. Also, a third-party web application can be created to receive notifications from the WCS server.

Configuration

Configure startup

By default, the WCS server starts in the dev mode.

To run the server with the 'production' profile, uncomment the following line in the /usr/local/FlashphonerWebCallServer/conf/wcs-manager.properties file:

```
-Dspring.profiles.active=production
```

In the production mode, support for HTTPS for REST is enabled.

CallApp application

http://localhost:9091/CallAppis an internal address available by default that receives all intermediate statuses of a call made via REST: RING, TRYING and so on.
Also, intermediate statuses of the RTMP stream associated with this call are sent:

- RTMP_STREAM_WAIT
- RTMP_STREAM_ACTIVE
- RTMP_CONNECTION_LOST
- RTMP_CONNECTION_FAILED

This address can be changed via WCS CLI. Se thedescription of the command line interfaceto get more information aboutapplication managementin WCS.

Enabling HTTPS

REST works via HTTP at the port 8081 (by default) and via HTTPS at the port 8444.

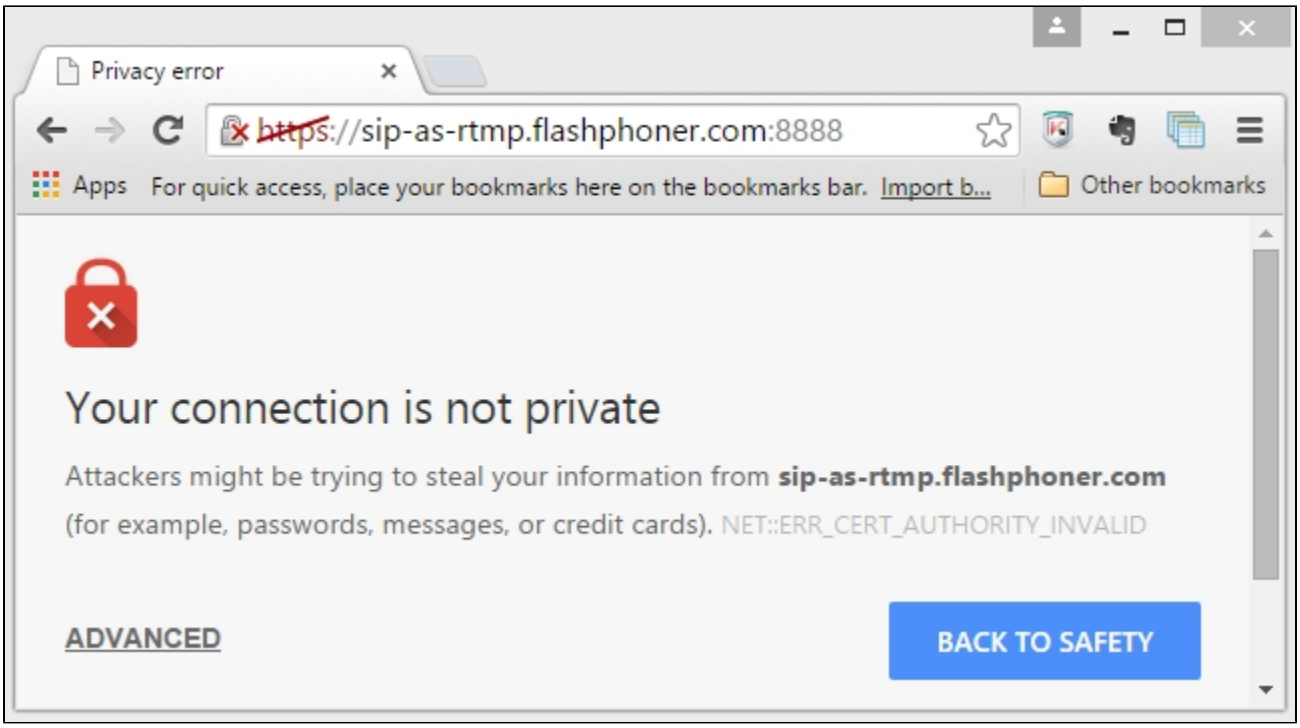
To enable HTTPS, start the server with the 'production' profile.

By default, WCS uses a self-signed SSL certificate.
To confirm security exception for this certificate:

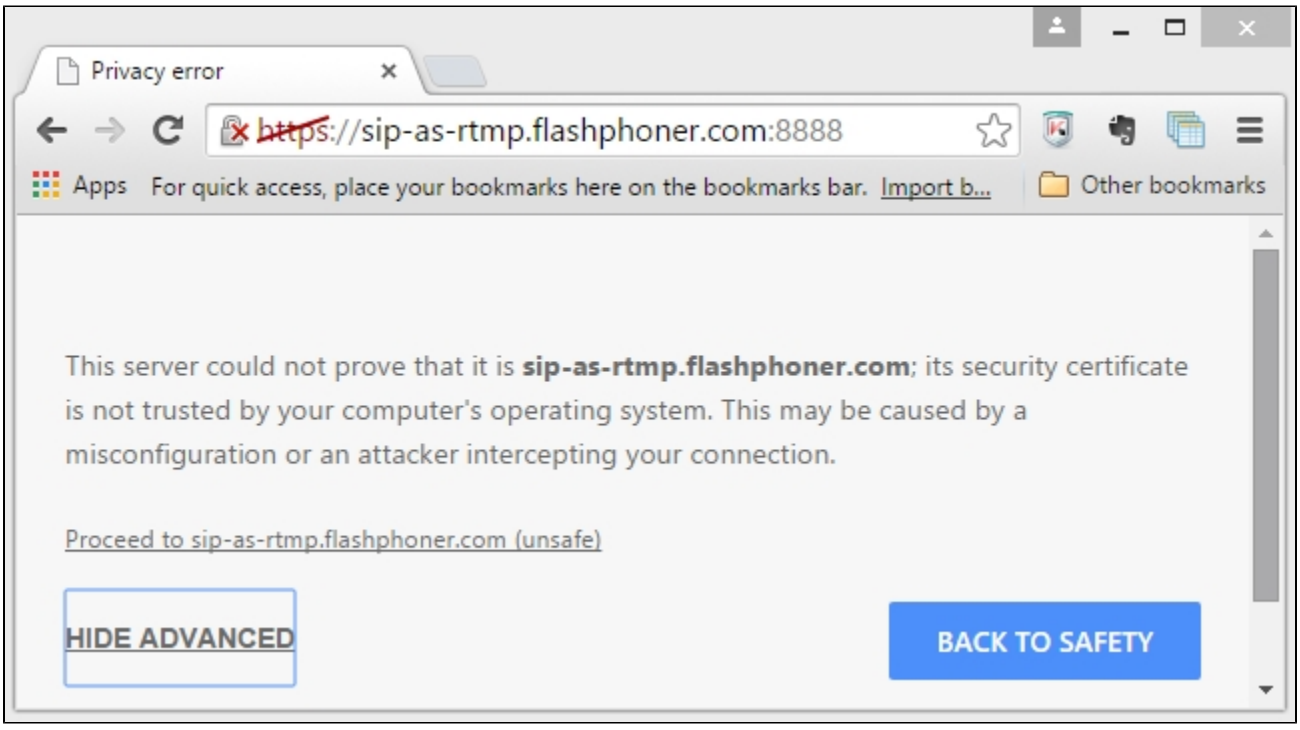
a) Open this URL

https://sip-as-rtmp.flashphoner.com:8444/

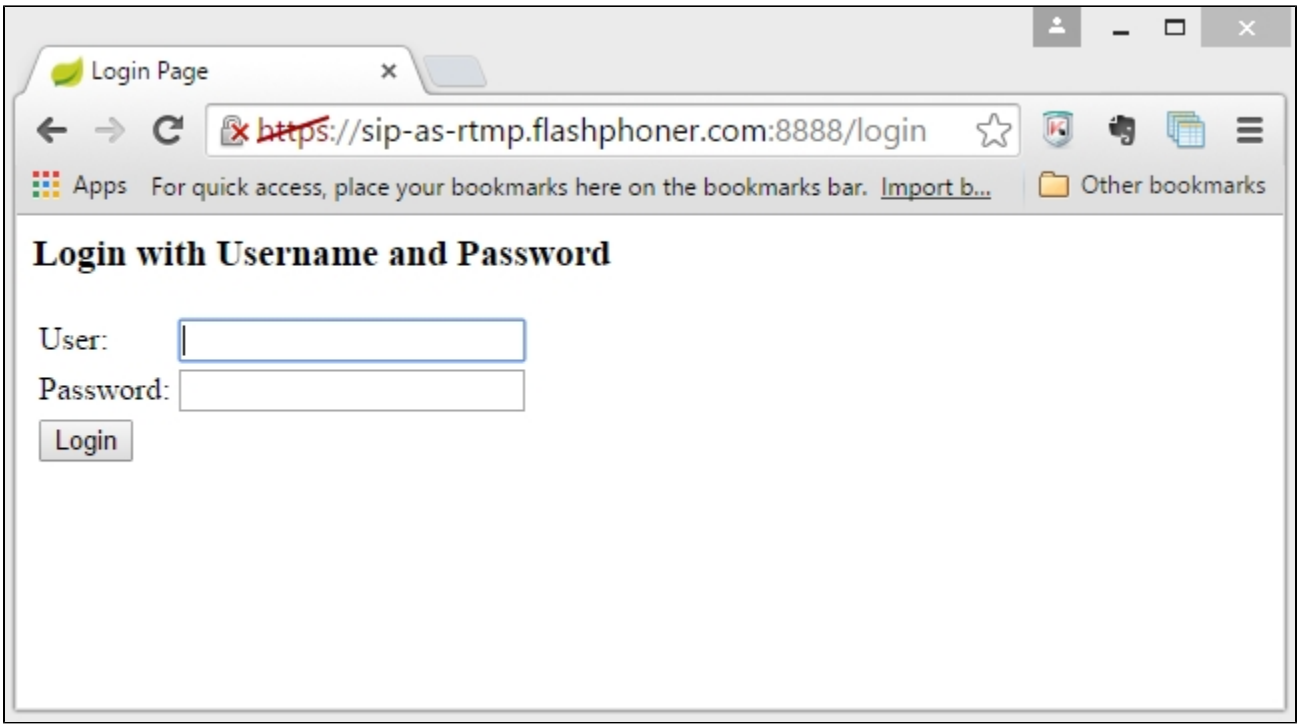
(where sip-as-rtmp.flashphoner.com - is the address of the WCS server)



b) Click 'ADVANCED'



c) Click 'Proceed'



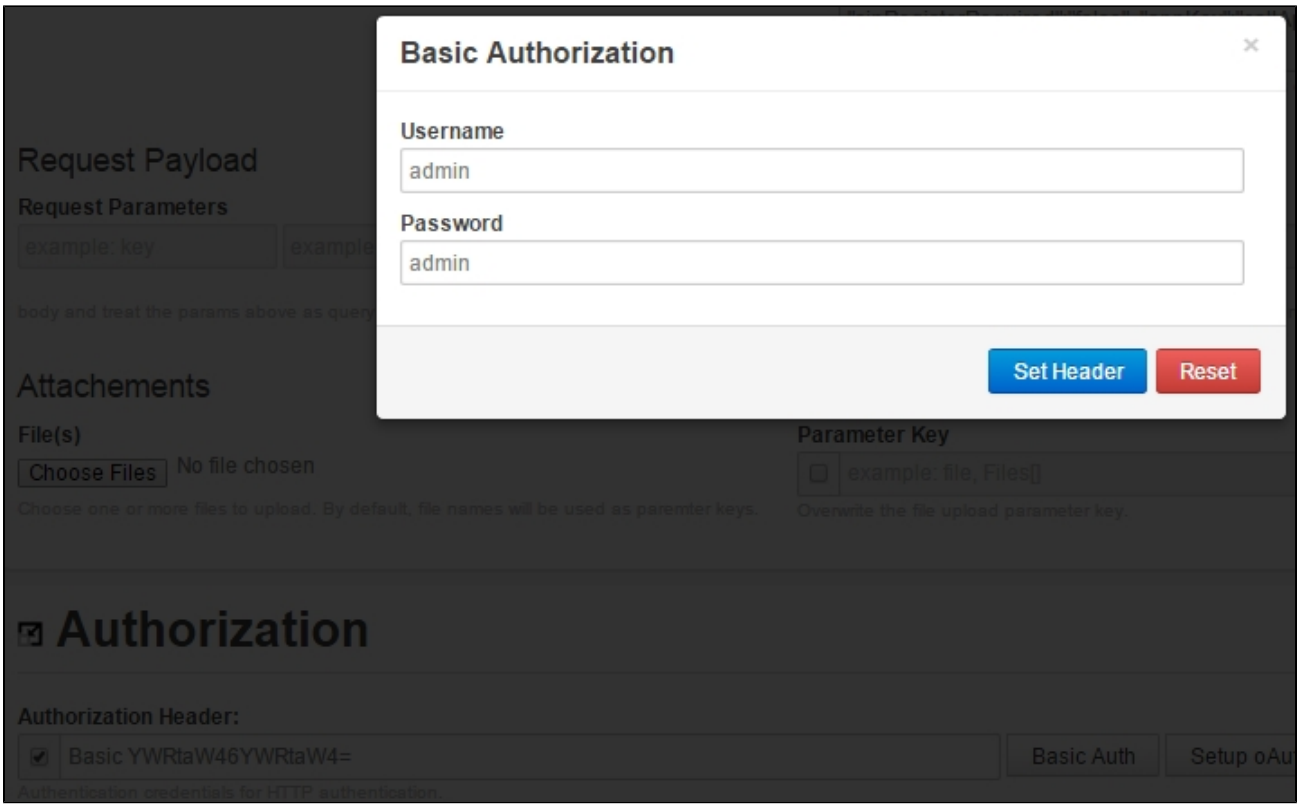
After that WCS self-signed certificate is imported to your browser and the HTTPS URL can be used for REST calls, for examplehttps://sip-as-rtmp.flashphoner.com:8444/rest-apil/call

Authentication

When the 'production' mode is on, each REST/HTTPS or REST/HTTP query requiresHTTP Basic Authentication.
Standard username and password are admin:admin.
You can change the password in WCS CLI. (See more about Command Line Interface here.)

In the REST Console, you can add authorization as follows

- click 'Basic Auth' in Authorization / Authorization Header,
- add admin:admin as username and password,
- click 'Set Header'



As a result, the authorization header is set.

Known issues

1. When republishing SIP as RTMP to Wowza servers and when receiving a stream from Wowza via HLS, a spectator can see freezes, short time non-synchronous playback.

Solution: enable transcoding on the server by setting in theflashphoner.propertiesfile the following parameter:

```
disable_streaming_proxy=true
```

2. When SIP call is redirecting to stream with SIP as Stream function, the audio only call stream does not play via WebRTC in a browser.

Solution: the audio only call stream should be played as audio only stream in a browser by explicitly constraints setting in player script when stream is created, for example

```
session.createStream({constraints:{audio:true,video:false}).play();
```