Pexip Infinity Version 10
Client REST API v2

Introduction

This guide describes the Pexip client REST API. It is designed for use by non-web-based, third-party voice/video applications that want to initiate or connect to conferences hosted on the Pexip Infinity platform.

We strongly recommend that web-based applications use the "PexRTC" JavaScript API instead.

This API specification is regularly evolving between versions of the Pexip Infinity platform. While we will attempt to maintain backward compatibility, there may be significant changes between versions.

Using the API

The prefix for all API calls is:

https://<node_address>/api/client/v2/conferences/<conference_alias>/

where **<node_address>** is the address of a Conferencing Node and **<conference_alias>** is an alias of the conference you are connecting to. Under this API path comes a sequence of response API calls, for example:

https://10.0.0.1/api/client/v2/conferences/meet_alice/request_token

All commands in the client API are authenticated with a token, which is presented by the Pexip Conferencing Node. The token has a validity lifetime, before the end of which it must be refreshed. The token is presented in a HTTP header entitled "token" on every HTTP request, except for the initial **request_token** request.

Unless otherwise specified, all payloads of requests and responses are JSON objects, Content-Type: application/json.

The responses have two fields, **status** and **result**:

- **status** is "success" if the command has been processed by Pexip, or "failure" if the command could not be processed. Note that this does not mean that the end result is success, only that the request has been received and processed.
- the **result** field indicates if the request was successful.

Summary of API requests and events

This section summarizes the requests and server-sent events that may be used, which are then described in more detail.

Client control requests

These REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/<request>

Request	GET/POST	Description
request_token	POST	Requests a new token from the Pexip Conferencing Node.
refresh_token	POST	Refreshes a token to get a new one.
release_token	POST	Releases the token (effectively a disconnect for the participant).

Conference control functions

These REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/<request>

Request	GET/POST	Description
conference_status	GET	Provides the status of the conference.
dial	POST	Dials out from the conference to a target endpoint.
lock / unlock	POST	Locks / unlocks the conference.
muteguests / unmuteguests	POST	Mutes / unmutes all guests on a conference.
disconnect	POST	Disconnects all conference participants, including the participant calling the function.
message	POST	Sends a message to all participants in the conference.
participants	GET	Returns the full participant list of the conference.

Participant functions

These participant REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/participants/<participant_uuid>/<request>

Request	GET/POST	Description
disconnect	POST	Disconnects a participant.
mute / unmute	POST	Mutes / unmutes a participant.
allowrxpresentation / denyrxpresentation	POST	Enables or disables a participant from receiving the presentation stream.
spotlighton / spotlightoff	POST	Enables or disables the "spotlight" on a participant.
unlock	POST	Lets a specified participant into the conference from the waiting room of a locked conference.

Request	GET/POST	Description
dtmf	POST	Sends RFC2833 DTMF digits to the participant.
calls	POST	Upgrades this connection to have a WebRTC or RTMP audio / video call element.

Call functions

These call REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/participants/<participant_uuid>/calls/<call_uuid>/<request>

Request	GET/POST	Description
ack	POST	Starts media for the specified call (WebRTC calls only).
disconnect	POST	Disconnects the specified call.
dtmf	POST	Sends RFC2833 DTMF digits to the conference. Only applies to gateway calls.

Server-sent events

To subscribe, open an HTTP connection to:

https://<node_address>/api/client/v2/conferences/<conference_alias>/events?token=<token_id>

Event	Description
presentation_start	Marks the start of a presentation, and includes the information on which participant is presenting.
presentation_stop	The presentation has finished.
presentation_frame	A new presentation frame is available.
participant_create	A new participant has joined the conference.
participant_update	A participant's properties have changed.
participant_delete	A participant has left the conference.
participant_sync_begin / participant_sync_end	These two messages start and end the sending of the complete participant list.
conference_update	Conference properties have been updated.
message	A chat message has been broadcast to the conference.
stage	An update to the "stage layout" is available. This declares the order of active speakers, and their voice activity.
call_disconnected	Sent when a child call has been disconnected.
disconnect	Sent when the participant is being disconnected from the Pexip side.

Client control requests

This section describes in detail the requests that may be used to initiate and manage a connection to a Conferencing Node.

These REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/<request>

request_token

This POST requests a new token from the Pexip Conferencing Node.

Request example:

```
{"display_name": "Alice"}
```

Request fields:

display_name	The name by which this participant should be known.

Response example:

```
{"status": "success", "result":
{"token": "SE9TVAltZ...etc...zNiZjlmNjFhMTlmMTJiYTE%3D",
"expires": "120",
"participant_uuid": "2c34f35f-1060-438c-9e87-6c2dffbc9980",
"display_name": "Alice",
"stun": [{"url": "stun:stun.l.google.com:19302"}],
"analytics_enabled": true,
"version": {"pseudo_version": "25010.0.0", "version_id": "10"},
"role": "HOST",
"service_type": "conference",
"chat_enabled": true,
"current_service_type": "conference"}}
```

This result contains the token (abridged in the above example) to use to authenticate all future requests, and an expiry time (in seconds) after which this token becomes invalid. The full list of fields in the result is as follows:

token	The authentication token for future requests.
expires	Validity lifetime in seconds. Use refresh_token to obtain an updated token.
participant_uuid	The uuid associated with this newly created participant. It is used to identify this participant in the participant list.
version	The version of the Pexip server being communicated with.
role	Whether the participant is connecting as a "HOST" or a "GUEST".
chat_enabled	true = chat is enabled; false = chat is not enabled.
service_type	Either "conference" or "gateway" depending on whether this is a VMR or a gateway call.
stun	STUN server configuration from the Pexip Conferencing Node.
display_name	Echoes the display name in the request.
analytics_enabled	Whether the **Automatically send deployment and usage statistics to Pexip** global setting has been enabled on the Pexip installation.
current_service_type	The service type this user is connecting into. May be "conference" or "gateway" as for **service_type** if directly connecting in. May also be "waiting_room" if waiting to be allowed to join a locked conference, or "ivr" if on the PIN entry screen.

PIN protected conferences

If the conference is PIN-protected, the PIN must be specified in a "pin" HTTP header. If the PIN is required but is incorrect or missing, a "403 Forbidden" error is returned. The "pin" field in the response specifies whether a PIN is required for hosts, and a "guest_pin" field in the response specifies whether a PIN is required for guests.

Virtual Receptions

If the conference is a Virtual Reception, a "403 Forbidden" error is returned, with a "conference_extension" field. This field is either "standard" for a standard Virtual Reception, or "mssip" for a Virtual Reception towards Microsoft Lync conferences. To join the target room, a second **request_token** request must be made, but with a **conference_extension** field in the request JSON, which contains the alias of the target conference.

refresh_token

This POST request refreshes a token to get a new one.

Request: empty.

Example response:

```
{"status": "success", "result":
{"token": "SE9TVAltZ...etc...jQ4YTVmMzM3MDMwNDFlNjI%3D",
"expires": "120"}}
```

Fields are:

token	The new authentication token for future requests.
expires	Validity lifetime in seconds.

release_token

This POST request releases the token (effectively a disconnect for the participant).

Request: empty.

Response: should be ignored.

Conference control functions

This section describes in detail the requests that may be used to manage an existing conference.

These REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/<request>

conference_status

This GET request provides the status of the conference. Currently, the only conference properties available are the lock status of the conference, and whether guests are muted. For example:

```
{"status": "success", "result":
{"guests_muted": false, "locked": false}}
```

dial

This POST request dials out from the conference to a target endpoint. This function is only available to conference hosts.

Request example:

```
{"role": "GUEST", "destination": "bob@example.com", "protocol": "sip"}
```

Request fields:

role	Role can be "GUEST" or "CHAIR" (for host).
destination	The target address to call.
protocol	The protocol to use: "sip", "h323", "mssip" (for calls to Microsoft Lync), or "rtmp".
presentation_url	This additional parameter can be specified for RTMP calls to send the presentation stream to a separate RTMP destination.
streaming	Optional field. Set to "yes" if this is to be treated as a streaming participant for recording purposes. Default = "no".

Response example:

```
{"status": "success", "result": ["977fcd1c-8e3c-4dcf-af45-e536b77af088"]}
```

The response is an array of UUIDs of new participants, if dial-out was successfully initiated. In most cases the dial-out will only generate a single call and thus a single UUID in this array, however if Pexip forks the call there may end up being multiple UUIDs. Only one of these will be answered, however, and the rest will be disconnected.

The call UUIDs will appear as new participants immediately, with a "service_type" of "connecting". If the call is answered, the participant will be updated with a new "service_type", typically being "conference". The participant may also be deleted if the receiver rejects the call, or the call attempt times out in 30 seconds if not answered.

lock / unlock

These POST requests are used to lock or unlock the conference. When a conference is locked, participants waiting to join are held at a "waiting for host" screen. These settings are only available to conference hosts.

Request: empty.

Response: the result is true if successful, false otherwise.

muteguests / unmuteguests

These POST requests are used to mute or unmute all guests on a conference. When muted, no "guest" participants can speak unless they are explicitly unmuted. When unmuted, all guests on a conference can speak. These settings are only available to conference hosts.

Request: empty.

Response: the result is true if successful, false otherwise.

disconnect

This POST request disconnects all conference participants, including the participant calling the function. This setting is only available to conference hosts.

Request: empty.

Response: the result is true if successful, false otherwise.

message

This POST request sends a message to all participants in the conference.

Request example:

```
{"type": "text/plain", "payload": "Hello World"}
```

Request fields:

type	The MIME Content-Type, such as "text/plain".
payload	The contents of the message.

Response: the result is true if successful, false otherwise.

participants

This GET request returns the full participant list of the conference. See the description of the participant_create EventSource for more information.

Participant functions

Within a conference, operations can be performed on participants, if the client has host privileges.

These participant REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/participants/<participant_uuid>/<request>

where **<node_address>** is the Conferencing Node, **<conference_alias>** is an alias of the conference, and **<participant_uuid>** is the uuid of the participant you are controlling. Under this path comes the request, for example:

https://10.0.0.1/api/client/v2/conferences/meet_alice/participants/7f8bdd7f-2d39-4c3f-9236-3e95b21f21a8/disconnect

disconnect

This POST request disconnects a participant.

Request: empty

Response: the result is true if successful, false otherwise.

mute / unmute

These POST requests are used to mute or unmute a participant.

Request: empty.

Response: the result is true if successful, false otherwise.

allowrxpresentation / denyrxpresentation

These POST requests are used to enable or disable a participant from receiving the presentation stream. (Participants are enabled by default.)

Request: empty.

Response: the result is true if successful, false otherwise.

spotlighton / spotlightoff

These POST requests are used to enable or disable the "spotlight" on a participant.

The spotlight feature locks any spotlighted participants in the primary positions in the stage layout. When any participants have been spotlighted, the first one to be spotlighted has the main speaker position, the second one has the second position (leftmost small video, for example), and so on. All remaining participants are arranged by most recent voice activity, as is default.

Request: empty.

Response: the result is true if successful, false otherwise.

unlock

This POST request lets a specified participant into the conference from the waiting room of a locked conference.

Request: empty.

Response: the result is true if successful, false otherwise.

dtmf

This POST request sends RFC2833 DTMF digits to the participant.

Request example:

```
{"digits": "1234"}
```

Request fields:

digits	The DTMF digits to send.

Response: the result is true if successful, false otherwise.

calls

This POST request upgrades this connection to have an audio/video call element. There are two variants of this request, depending upon whether a WebRTC or RTMP call is to be established.

WebRTC

Request example to add a WebRTC element:

```
{"call_type": "WEBRTC", "sdp": "..."}
```

Request fields:

call_type	"WEBRTC" for a WebRTC call.
sdp	Contains the SDP of the sender.
present	Optional field. Contains "send" or "receive" to act as a presentation stream rather than a main audio/video stream.

Response example (WebRTC):

```
{"status": "success", "result": {
"call_uuid": "50ed679d-c622-4c0e-b251-e217f2aa030b",
"sdp": "..."}}
```

The response contains the SDP of the Pexip node, and a call_uuid. This call_uuid is used to control the call. The ack function must be called on this call_uuid in order to start media after the SDP has been exchanged and ICE has been completed.

RTMP

Request example to add an RTMP element:

```
{"call_type": "RTMP"}
```

Request fields:

call_type	"RTMP" for an RTMP call.
present	Optional field. Contains "send" or "receive" to act as a presentation stream rather than a main audio / video stream.
streaming	Optional field. Set to "true" if this is to be treated as a streaming participant for recording purposes.
bandwidth	Optional field. If supplied it provides a maximum incoming / outgoing bandwidth in kbps.

Response example (RTMP):

```
{"status": "success", "result": {
"call_uuid": "50ed679d-c622-4c0e-b251-e217f2aa030b",
"url": "rtmp://10.0.0.1:40002/pexip/50ed679d-c622-4c0e-b251-e217f2aa030b",
"secure_url": "rtmps://hostname.domain:40003/pexip/50ed679d-c622-4c0e-b251-
e217f2aa030b"}}
```

The response contains RTMP URLs that can be connected to by the client – both an insecure (rtmp://) and secure (rtmps://) variant. The RTMPS URL is only returned if a SIP TLS FQDN is configured for the Conferencing Node, and requires a valid TLS certificate to be installed on the Conferencing Node.

Call functions

Using the **call_uuid**, further operations can be undertaken on the calls as part of the nominated participant.

These call REST URIs take the format:

https://<node_address>/api/client/v2/conferences/<conference_alias>/participants/<participant_uuid>/calls/<call_uuid>/<request>

where **<node_address>** is the Conferencing Node, **<conference_alias>** is an alias of the conference, **<participant_uuid>** is the uuid of the participant, and **<call_uuid>** is the uuid of the call you are controlling. Under this path comes the request, for example:

https://10.0.0.1/api/client/v2/conferences/meet_alice/participants/7f8bdd7f-2d39-4c3f-9236-3e95b21f21a8/calls/c34f35f-1060-438c-9e87-6c2dffbc9980/disconnect

ack

This POST request starts media for the specified call (WebRTC calls only).

Request: empty.

Response: empty.

disconnect

This POST request disconnects the specified call.

Request: empty.

Response: the result is true if successful, false otherwise.

dtmf

This POST request sends RFC2833 DTMF digits to the conference. Only applies to gateway calls.

Request example:

```
{"digits": "1234"}
```

Response: the result is true if successful, false otherwise.

Server-sent events

Clients can subscribe to an HTTP EventSource which feeds events from the conference as they occur.

To subscribe, open an HTTP connection to:

https://<node_address>/api/client/v2/conferences/<conference_alias>/events?token=<token_id>

where **<node_address>** is the Conferencing Node, **<conference_alias>** is an alias of the conference, and **<token_id>** is the session token, for example:

https://10.0.0.1/api/client/v2/conferences/meet_alice/events?token=123456

This allows the token to be specified on the URI, since custom headers cannot be added to Event Sources in browsers today. However, if headers can be added this will be accepted too, and the query parameter will not be required.

Each event contains an event name, and some events may contain a payload of data, which is a JSON object.

presentation_start

This marks the start of a presentation, and includes the information on which participant is presenting.

Example data:

```
{"presenter_name": "Bob", "presenter_uri": "bob@example.com"}
```

presentation_stop

The presentation has finished.

Data: none

presentation_frame

A new presentation frame is available at:

https://<node_address>/api/client/v2/conferences/<conference_alias>/presentation.jpeg

Note that this URL requires the token to be present as a header or a query parameter in order to download the presentation frame.

Data: none

participant_create

A new participant has joined the conference.

The JSON object fields include:

call_direction	Either "in" or "out" as to whether this is an inbound or outbound call.
display_name	The display name of the participant.
encryption	"On" or "Off" as to whether this participant is connected via encrypted media.
has_media	Boolean indicating whether the user has media capabilities.
is_muted	Set to "YES" if the participant is administratively muted.
is_presenting	Set to "YES" if the participant is the current presenter.
is_streaming_conference	Boolean indicating whether this is a streaming participant.
protocol	The protocol with which the participant is connecting.
role	Either "chair" (host) or "guest".
rx_presentation_policy	Set to "ALLOW" if the participant is administratively allowed to receive presentation, or "DENY" if disallowed.
service_type	The service type: • "connecting" for a dial-out participant that has not been answered • "waiting_room" if waiting to be allowed to join a locked conference • "ivr" if on the PIN entry screen • "conference" if in the VMR • "gateway" if it is a gateway call
spotlight	A Unix timestamp of when this participant was spotlighted, if spotlight is used.
start_time	A Unix timestamp of when this participant joined (UTC).
uuid	The UUID of this participant, to use with other operations.
uri	The URI of the participant.
vendor	The vendor identifier of the browser/endpoint with which the participant is connecting.

Example data:

```
{"api_url": "/participants/50b956c8-9a63-4711-8630-3810f8666b04"
"call_direction": "in"
"display_name": "Alice"
"encryption": "On"
"has_media": false
"is_muted": "NO"
"is_presenting": "NO"
"is_streaming_conference": false
"local_alias": "meet.alice"
"overlay_text": "Alice"
"presentation_supported": "NO"
"protocol": "api"
"role": "chair"
"rx_presentation_policy": "ALLOW"
"service_type": "conference"
```

```
"spotlight": 0
"start_time": 1441720992
"uri": "Infinity_Connect_10.44.21.35"
"uuid": "50b956c8-9a63-4711-8630-3810f8666b04"
"vendor": "Pexip Infinity Connect/2.0.0-25227.0.0 (Windows NT 6.1; WOW64) nwjs/0.12.2
Chrome/41.0.2272.76"}
```


participant_update

A participant's properties have changed.

Data: a full JSON object is supplied, as for participant_create.

participant_delete

A participant has left the conference.

Data: the JSON object contains the UUID of the deleted participant, e.g:


```
{"uuid": "65b4af2f-657a-4081-98a8-b17667628ce3"}
```


participant_sync_begin / participant_sync_end

At the start of the EventSource connection, these two messages start and end the sending of the complete participant list in the form of participant_create events. This allows a participant that has been temporarily disconnected to re-sync the participant list.

conference_update

Conference properties have been updated. Currently, the only conference properties available are the lock status of the conference, and whether guests are muted. For example:


```
{"locked": false, "guests_muted": false}
```


message

A chat message has been broadcast to the conference.

Data: an object containing the following fields:

origin	Name of the sending participant.
uuid	UUID of the sending participant.
type	MIME content-type of the message, usually text/plain.
payload	Message contents.

Example data:


```
{"origin": "Alice",
"type": "text/plain",
"payload": "Hello World",
"uuid": "eca55900-274d-498c-beba-2169aad9ce1f"}
```

stage

An update to the "stage layout" is available. This declares the order of active speakers, and their voice activity.

Data: an array of objects per active participant. Each participant has the following fields:

participant_uuid	The UUID of the participant.
stage_index	The index of the participant on the "stage". 0 is most recent speaker, 1 is the next most recent etc.
vad	Audio speaking indication. 0 = not speaking, 100 = speaking.

Example data:

```
[
{"stage_index": 0,
"participant_uuid": "a0196175-b462-48a1-b95c-f322c3af57c1",
"vad": 0},
{"stage_index": 1,
"participant_uuid": "65b4af2f-657a-4081-98a8-b17667628ce3",
"vad": 0}
]
```

call_disconnected

This is sent when a child call has been disconnected (e.g. when a screensharing child call has been closed if presentation has been stolen by another participant).

Data: contains both the UUID of the child call being disconnected, and the reason for the disconnection if available, e.g.:

```
{"call_uuid": "50ed679d-c622-4c0e-b251-e217f2aa030b",
"reason": "API initiated participant disconnect"}
```

disconnect

This is sent when the participant is being disconnected from the Pexip side.

Data: the reason parameter contains a reason for this disconnection, if available, e.g.:

```
{"reason": "API initiated participant disconnect"}
```

Changelog

Changes since v9.1:

- WebRTC and Lync multiple participant records for the same participant are no longer seen; you now receive a single record per participant, so there is longer a participant_id value.
- DTMF tones can be sent to a specific participant.
- request_token can be used to access a Pexip Virtual Reception.
- The dial out command now returns an array of UUIDs.

More information

Questions about this API should be directed to support@pexip.com.