xroom.app backend API manual

(updated 2021-02-06, version code 2)

Table of contents

Prerequisites

Authentication

The platform currently uses SHA256 HMAC authentication algorithm. Three HTTP request headers participate in the authentication process: `x-random` is a salt (32 bytes minimum), `x-auth-id` is your internal identifier and `x-auth-key` is the hash computed based on the two above and your secret key. Both ID and secret can be found in the dashboard, secret can also be regenerated at any moment.

The hash is computed directly from the salt using your API secret key. For your convenience in the dashboard there is a hash debugging tool: https://my.xroom.app/en/api-manager.

Versioning

You may send a header `x-api-version` indicating desired API version code. Starting from version 2 this header is obligatory.

API wrappers

We have started adding sample API wrappers for various programming languages: https://github.com/xroom-app/api-wrappers.

Data exchange

Requests are sent as HTTP POST with JSON payload, thus `Content-Type: application/json` header is expected. Endpoint URL is composed of a root URL and a collection name, for example `https://api.xroom.app/api/room`. API method name to be called is passed directly in the payload:

```
{
    "method": "init",
    "data":
     {
          "id": "my-test-room",
          "key": "my-room-key"
     }
}
```

Data is returned as

```
{
    "isError": false,         // boolean showing if any error was intercepted
    "errMsg": null,           // possible execution error message
    "data":                   // data returned by the method itself
    {
        ...
    }
}
```

Collections

room (https://api.xroom.app/api/room)

Room identifier is combined from the domain name and a room name, e.g. chat.org/my-room
For rooms it is important to use the server (API host) where your room is located. xroom is built in a way information about rooms in one country is not copied to another country without necessity. This is both helpful techwise and makes it easier to comply with various local laws.

Method	Description	Input data	Output data

init	Initialize a room	{
		`id: str` — room identifier
		`type: ?uint` — room type code, see below, 1 by default
		`lock: ?bool` — lock flag, false by default
		`key: ?str` — room password
		`addHostKey: ?bool` — request a predefined host key
		}
			{
			`id: str` — room identifier
			`type: uint` — room type code, see below
			`isLocked: bool` — room lock flag
			`key: ?str` — room password
			`hostKey: ?str` — a predefined host key
			}

destroy	Destroy an empty room	{
		`id: str` — room identifier
		`forceKick: ?bool` — pass true to kick everyone before
		}
			`bool` — whether operation was successful or not

kickOut	Kick out a user	{
		`id: str` — room identifier
		`peerId: str` — peer identifier, pass "all" to kick everyone
		}
			—

setLock	Set room locking	{
		`id: str` — room identifier
		`lock: bool` — boolean lock flag
		}
			—

setPassword	Set room password	{ `id: str` — room identifier `password: str|null` — password string, pass null to reset	—

		} 	

list	List active rooms	{
		  `domain: ?str` — domain to filter by
		}
			{
			  [domain-1]: {
			  [domain-1/room-1]: ?RoomObject,
			  ...
			  },
			  [domain-2]: {
			  [domain-2/room-1]: ?RoomObject,
			  ...
			  },
			  ...
			}
			
			RoomObject: {
			  options: {
			  type: uint,
			  isLocked: bool,
			  password: str
			  },
			  clients: [PeerData]
			}
			
			PeerData: {
			  id: str,
			  type: uint
			}

booking (https://api.xroom.app/api/booking)

Booking represents a reservation of a specific room until a specific time in the future. When a room is booked it is not possible to initialize it from the interface while the booking is valid. If let's say a room is booked for 10:00 and it is allowed for the speaker to be delayed up to 15 minutes an expiration time for this reservation has to be set to 10:15.

Method	Description	Input data	Output data

add	Add a booking	`domain: str` — domain name
		`roomName: str` — room name
		`startTime: time` — meeting starting time, in JS format
		`meetingOptions: (`
		`duration: uint` — meeting time in minutes
		`lock?: bool (false by default)` — lock room, only host is able to enter after meeting started
		`type?: 1 | 2 (1 by default)` — meeting room type, 1 - conference, 2 - webinar
		`capacity?: uint (1..256)` — room capacity
		`key?: str` — password to protect room with
		`addHostKey?: bool (false by default)` — generate host key for the room and return in response
		)
			{
			`id: uint` — booking id
			`hostKey?: str` — your host key (will exist if 'addHostKey' = true)
			}

remove	Remove a booking	`id: uint` — booking id	—

update	Update a booking	`id: uint` — booking id
		`startTime: time` — meeting start time, in JS format
		`meetingOptions: (`
		`duration: uint` — meeting time in minutes
		`lock?: bool (false by default)` — lock room, only host is able to enter after meeting started
		`type?: 1 | 2 (1 by default)` — meeting room type, 1 - conference, 2 - webinar
		`capacity?: uint (1..256)` — room capacity
		`key?: str` — password to protect room with
		`addHostKey?: bool (false by default)` — generate host key for the room and return in response
		)
			{
			`hostKey?: str` — your host key (will exist if 'addHostKey' = true)
			}

list	List bookings	`domain: str` — domain name	`[BookingObject]`
			
			`BookingObject: {`
			`  id: uint` – booking id
			`  domainName: str` — domain name
			`  roomName: str` — room name
			`  startTime: time` — meeting start time,
			`  meetingOptions: {`
			`  duration: uint` – duration in minutes
			`  type: 1 | 2` – room type
			`  isLocked: bool` – room is locked
			`  capacity: uint` – room capacity
			`  password?: str` – room password
			`  hostKey?: str` – room host key
			`  }`
			`}`

get	Read a booking	`id: uint` – booking id	`BookingObject`

webhook (https://api.xroom.app/api/webhook)

Webhooks allow you to trigger an external listener on a specific event. Currently the only supported event is a room creation.

Method	Description	Input data	Output data

add	Add a webhook	`domain: str` — domain name
		`type: enum` — webhook type: 'url' or 'robot'
		`roomName: ?str` — room name
		`url: ?str` — webhook URL for type 'url'
		`robotId: ?uint` — robot identifier for type 'robot'
			{
			`id: uint` — webhook identifier
			`validated: bool` — validation flag ('robot' type has 'true')
			}

remove	Remove a webhook	`id: uint` — webhook id	—

list	List webhooks	`domain: str` — domain name
		`token: ?str` — possible search token
		`validated: ?bool` — a validation flag to filter by
			`[WebhookObject]`
			
			`WebhookObject: {`
			`id: uint` — webhook identifier
			`domain: str` — domain name
			`roomName: ?str` — room name
			`url: ?str` — webhook URL in case of type 'url'
			`robotId: ?uint` — robot identifier in case of type 'robot'
			}

validate	Validate a webhook	`id: uint` — webhook id	{
			`validated: bool` — resulting validation flag
			}

Webhook validation request is an HTTP POST with JSON payload of `{event: "validation", signThis: str}` and an expected response is a string of `signThis` signed with domain's `webhookKey` using SHA256 algorithm.

Example:

```
{event: "validation", signThis: "c9002b50c6911daa18854e1ec25d5b47"}
+ secret "54ed8ea9722c38499ea8dbbeca19762e"
=> "b31ecd741c86f3abe01d93ebceb19b73cc2595bb5f5f3d4ba8328b3c37e6698b"
```

Appendix A: constant types

room types

Code	Type	Description
1	conference	All participants are equal, can talk and chat to each other
2	webinar	A host streams, the rest are watching and can chat with each other.

peer types

Code	Type
1	Conference host
2	Conference user
3	Webinar host
4	Webinar viewer

Appendix B: use case “booking a private room”

You want to be able to book a room for an event via the API so that no one could take it over while the corresponding booking is active. When the time comes you want to activate that room but need to distinguish its host from everyone else. Here’s what you need to do:

0. Configure your domain in the back office if you need to prohibit non-API room creation or to change plugins accessibility.
1. Call `booking/add` to book your room in advance.
2. When time comes, call `room/init` explicitly passing `addHostKey` parameter as true.
3. You will get `hostKey` in the response. Now the room is created but anyone joining it directly will not become a host.
4. Construct a host URL as `https://[room-id]?host-key=[hostKey]` and for the rest it’s just `https://[room-id]`.

If you need to set a password for participants simply pass the `key` parameter in step 2.

Appendix C: webhook events and payloads

Payload field “event”	Description
validation	Webhook validation request. See details above in the webhook API section.

roomCreated	<pre>{
	  webhookId: uint,
	  event: 'roomCreated'
	  domain: str
	  payload: {
	  room: {
	  name: str,
	  type: 1 | 2, // see room types above
	  capacity: uint,
	  isLocked: bool,
	  password?: str,
	  hostKey?: str
	  }
	  }
	}</pre>

roomRemoved	<pre>{
	  webhookId: uint,
	  event: 'roomRemoved',
	  domain: str,
	  payload: {
	  room: {
	  name: str,
	  }</pre>

	} }

clientAdded	{
	webhookId: uint,
	event: 'clientAdded'
	domain: str
	payload: {
	room: {
	name: str
	},
	client: {
	id: str,
	type: 1 | 2,
	}
	}
	}

clientsRemoved	{
	webhookId: uint,
	event: 'clientsRemoved'
	domain: str
	payload: {
	room: {
	name: str
	},
	clients: [{
	id: str,
	type: 1 | 2,
	}]
	}
	}