## ROCKET.CHAT

Rocket.Chat is a Web Chat Server, developed in JavaScript, using the [Meteor](https://www.meteor.com/install) full stack framework.

It is a great solution for communities and companies wanting to privately host their own chat service or for developers looking forward to build and evolve their own chat platforms.

Rocket.Chat is an open source communication hub that enables banks, NGOs, startups, and governmental organizations to have their own chat tool, customize its look and feel, choose their users, and securely manage data.

Users can set up Rocket.Chat on cloud or by hosting their own servers on-premises.

Rocket.Chat can be forked on Github and become a new platform altogether, thanks to its code being open source and thus openly available to the community.

Key features include free audio and video conferencing, guest access, screen and file sharing, LiveChat, LDAP Group Sync, two-factor authentication (2FA), E2E encryption, SSO, dozens of OAuth providers and unlimited users, guests, channels, messages, searches and files. Customization is enabled with the option of adding or removing features and selecting your own integrations, plugins, themes, mobile and desktop apps.

**Features**

* BYOS (Bring Your Own Server)
* Multiple Rooms
* Direct Messages
* Private Groups
* Public Channels
* Desktop Notifications
* Mentions
* Avatars
* Markdown
* Emojis
* Custom Emojis
* Reactions
* One touch Geolocation
* TeX Math Rendering - inline math typesetting
* Media Embeds
* Link Previews
* Sent Message Edit and Deletion
* Transcripts / History
* File Upload / Sharing
* Scalable file sharing - S3 uploads with CDN downloads
* Full text search
* Global search (from all channels/rooms at once)
* Live chat / Messaging call center
* LDAP Authentication
* CAS 1.0, 2.0 support for educational institutions and hosting providers worldwide
* Support for Okta SSO through SAML v2
* I18n - Supports 22 Languages
* Hubot Friendly
* (Beta) Face to Face Video Conferencing (aka WebRTC )
* (Beta) Multi-users Video Group Chat
* (Beta) Jitsi integration
* Audio calls
* Multi-users Audio Conference
* Screen sharing
* Drupal 7.x and 8.x Plug-in (both stable and development flavours) ([download](https://www.drupal.org/project/rocket_chat) and [source code](https://git.drupal.org/project/rocket_chat.git) )
* XMPP bridge ([try it](https://open.rocket.chat/channel/general))
* REST APIs
* Remote Video Monitoring
* Native real-time APIs for Microsoft C#, Visual Basic, F# and other .NET supported languages ([Get it!](https://www.nuget.org/packages/Rocket.Chat.Net/0.0.12-pre))
* API access from [Perl](https://metacpan.org/pod/Net::RocketChat) and [Java](https://github.com/baloise/rocket-chat-rest-client) (community contributions)
* Chat-ops powered by Hubot: scalable horizontal app integration (early access)
* Massively scalable hosting and provisioning (beta testing now)
* Native Cross-Platform Desktop Application [Windows, macOS, or Linux](https://rocket.chat/)
* Mobile app for iPhone, iPad, and iPod touch [Download on App Store](https://geo.itunes.apple.com/us/app/rocket-chat/id1148741252?mt=8)
* Mobile app for Android phone, tablet, and TV stick [Available now on Google Play](https://play.google.com/store/apps/details?id=chat.rocket.android)
* Available on [Cloudron Store](https://cloudron.io/appstore.html" \l "chat.rocket.cloudronapp)

# **Realtime API**

Our real-time API is composed of two elements: [Method Calls](/api/realtime-api/method-calls) and [Subscriptions](/api/realtime-api/subscriptions). Both of them are supported directly in the websocket connection.

To make it possible to have everything working on the same connection we use RPC with the following format.

{

"msg": "type-of-communication",

"id": "unique-id",

... // per call defined data

}

The type of communication is defined according to the call:

* [Method Calls](/api/realtime-api/method-calls): method
* [Subscriptions](/api/realtime-api/subscriptions): sub

Please note, the server will send you "ping" and you must respond with "pong" otherwise the server will close the connection.

Before requesting any method / subscription you have to send a connect message:

{

"msg": "connect",

"version": "1",

"support": ["1"]

}

# Resources

* A basic example script that uses the 'ddp' NodeJS package to subscribe to the Realtime-API stream of a Group/Channel here <https://github.com/jszaszvari/rocketchat-ddp-listener>
* [Rocket.Chat.RealTime.API.RxJS](https://github.com/inf3cti0n95/Rocket.Chat.RealTime.API.RxJS) Abstraction for Utilizing [Rocket.Chat](https://rocket.chat/)'s [Realtime API](https://rocket.chat/docs/developer-guides/realtime-api) Methods with [RxJS](http://reactivex.io/rxjs/). <https://github.com/inf3cti0n95/Rocket.Chat.RealTime.API.RxJS>

Method calls are used to trigger actions based on the passed data. The response to any method call is completely asynchronous and there's no way to guarantee the order on the fulfillment of the calls. Because of that, it is really important that a unique-id is used on the method call since the same ID will be used on the response so the client may know the result of a call.

# **Livechat Realtime API**

This API is intended to be used for having a Livechat conversation;

* First of all you need to generate a visitor token (any random string);
* Call [livechat:getInitialData](https://docs.rocket.chat/api/realtime-api/livechat-api/getinitialdata) passing visitorToken as first argument, the response will be an object containing a Livechat [configuration object](https://docs.rocket.chat/api/realtime-api/livechat-api/getinitialdata#response) with following properties:

|  |  |  |
| --- | --- | --- |
| Field | Type | Description |
| enabled | Boolean | If whether Livechat is enabled for that server or not |
| title | String | The Livechat widget title |
| color | Hexadecimal | The hexadecimal color of the Livechat widget title bar when the Livechat is online |
| registrationForm | Boolean | If the registration form should be displayed or not. |
| room | Object | The current conversation room for the current guest user |
| visitor | Object | The current guest user |
| triggers | Array | Array of Livechat triggers. |
| departments | Array | Array of Livechat departments. |
| allowSwitchingDepartments | Boolean | If client-side department switching is allowed |
| online | Boolean | If there are Livechat agents online |
| offlineColor | Hexadecimal | The hexadecimal color of the Livechat widget title bar when the Livechat is offline |
| offlineMessage | String | The message that will be displayed on the Livechat offline form |
| offlineSuccessMessage | String | The message that will be displayed after send a message using the offline form |
| offlineUnavailableMessage | String | The message that will be displayed when the Livechat is offline |
| displayOfflineForm | Boolean | If the offline form will be displayed when the Livechat is offline |
| videoCall | Boolean | If the VideoCall feature is available |
| conversationFinishedMessage | Boolean | The system message that will be send to the current guest user when the conversation is closed |
| nameFieldRegistrationForm | Boolean | If the name field will be displayed on Livechat registration form |
| emailFieldRegistrationForm | Boolean | If the email field will be displayed on Livechat registration form |
| offlineTitle | String | The title of the widget when the Livechat is offline |
| language | String | The default user language to be set in the Livechat widget |
| transcript | Boolean | If the Livechat widget will ask the current guest user if they would like a transcript after the conversation is closed |
| transcriptMessage | String | The message to be displayed when asking about transcript |
| agentData | Object | The current agent attending the chart |

* Call [livechat:registerGuest](https://docs.rocket.chat/api/realtime-api/livechat-api/registerguest) to register guest and get the response containing the visitor's data.
* Before sending the first message you have to generate a random room \_id;
* Now you can send messages to method [sendMessageLivechat](https://docs.rocket.chat/api/realtime-api/livechat-api/sendmessagelivechat)​
* Subscribe to: [stream-room-messages](https://docs.rocket.chat/api/realtime-api/subscriptions/stream-room-messages) and stream-livechat-room
* Get agent info by calling livechat:getAgentData

# **REST API**

## Miscellaneous Information

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/info | Information about the Rocket.Chat server. | [Link](/api/rest-api/methods/miscellaneous/info) |
| /api/v1/directory | Search by all users and channels available on server. | [Link](/api/rest-api/methods/miscellaneous/directory) |
| /api/v1/shield.svg | Gets the shield svg(badge) to add in your website. | [Link](/api/rest-api/methods/miscellaneous/shield-svg) |
| /api/v1/spotlight | Searches for users or rooms that are visible to the user. | [Link](/api/rest-api/methods/miscellaneous/spotlight) |
| /api/v1/statistics | Statistics about the Rocket.Chat server. | [Link](/api/rest-api/methods/miscellaneous/statistics) |
| /api/v1/statistics.list | Selectable statistics about the Rocket.Chat server. | [Link](/api/rest-api/methods/miscellaneous/statistics-list) |

## Assets

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/assets.setAsset | Set an asset image by name. | [Link](/api/rest-api/methods/assets/setasset) |
| /api/v1/assets.unsetAsset | Unset an asset by name | [Link](/api/rest-api/methods/assets/unsetasset) |

## AutoTranslate

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/autotranslate.getSupportedLanguages | Get the supported languages by the autotranslate. | [Link](/api/rest-api/methods/autotranslate/getsupportedlanguages) |
| /api/v1/autotranslate.saveSetttings | Save some settings about autotranslate. | [Link](/api/rest-api/methods/autotranslate/savesettings) |
| /api/v1/autotranslate.translateMessage | Translate the message. | [Link](/api/rest-api/methods/autotranslate/translatemessage) |

## Authentication

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/login | Authenticate with the REST API. | [Link](/api/rest-api/methods/authentication/login) |
| /api/v1/login | Authenticate with google. | [Link](/api/rest-api/methods/authentication/google) |
| /api/v1/login | Authenticate with facebook. | [Link](/api/rest-api/methods/authentication/facebook) |
| /api/v1/login | Authenticate with twitter. | [Link](/api/rest-api/methods/authentication/twitter) |
| /api/v1/logout | Invalidate your REST API authentication token. | [Link](/api/rest-api/methods/authentication/logout) |
| /api/v1/me | Displays information about the authenticated user. | [Link](/api/rest-api/methods/authentication/me) |

## Users

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/users.presence | Gets all connected users presence. | [Link](/api/rest-api/methods/users/presence) |
| /api/v1/users.create | Create a new user. | [Link](/api/rest-api/methods/users/create) |
| /api/v1/users.createToken | Create a user authentication token. | [Link](/api/rest-api/methods/users/createtoken) |
| /api/v1/users.deactivateIdle | Deactivate idle users. | [Link](/api/rest-api/methods/users/deactivateidle) |
| /api/v1/users.delete | Deletes an existing user. | [Link](/api/rest-api/methods/users/delete) |
| /api/v1/users.deleteOwnAccount | Deletes your own user. | [Link](/api/rest-api/methods/users/deleteownaccount) |
| /api/v1/users.forgotPassword | Send email to reset your password. | [Link](/api/rest-api/methods/users/forgotpassword) |
| /api/v1/users.generatePersonalAccessToken | Generate Personal Access Token. | [Link](/api/rest-api/methods/users/generatepersonalaccesstoken) |
| /api/v1/users.getAvatar | Gets the URL for a user's avatar. | [Link](/api/rest-api/methods/users/getavatar) |
| /api/v1/users.getPersonalAccessTokens | Gets the user's personal access tokens. | [Link](/api/rest-api/methods/users/getpersonalaccesstokens) |
| /api/v1/users.getPreferences | Gets all preferences of user. | [Link](/api/rest-api/methods/users/get-preferences) |
| /api/v1/users.getPresence | Gets the online presence of a user. | [Link](/api/rest-api/methods/users/getpresence) |
| /api/v1/users.getStatus | Gets the user's status. | [Link](/api/rest-api/methods/users/getstatus) |
| /api/v1/users.getUsernameSuggestion | Gets a suggestion a new username to user. | [Link](/api/rest-api/methods/users/getusernamesuggestion) |
| /api/v1/users.info | Gets a user's information, limited to the caller's permissions. | [Link](/api/rest-api/methods/users/info) |
| /api/v1/users.list | All of the users and their information, limited to permissions. | [Link](/api/rest-api/methods/users/list) |
| /api/v1/users.regeneratePersonalAccessToken | Regenerate a user personal access token. | [Link](/api/rest-api/methods/users/regeneratepersonalaccesstoken) |
| /api/v1/users.register | Register a new user. | [Link](/api/rest-api/methods/users/register) |
| /api/v1/users.removeOtherTokens | Remove all other user tokens | [Link](/api/rest-api/methods/users/removeothertokens) |
| /api/v1/users.removePersonalAccessToken | Remove a personal access token. | [Link](/api/rest-api/methods/users/removepersonalaccesstoken) |
| /api/v1/users.requestDataDownload | Request users download data. | [Link](/api/rest-api/methods/users/requestdatadownload) |
| /api/v1/users.resetAvatar | Reset a user's avatar | [Link](/api/rest-api/methods/users/resetavatar) |
| /api/v1/users.setAvatar | Set a user's avatar | [Link](/api/rest-api/methods/users/setavatar) |
| /api/v1/users.setPreferences | Set user's preferences | [Link](/api/rest-api/methods/users/set-preferences) |
| /api/v1/users.setStatus | Set the user's status | [Link](/api/rest-api/methods/users/setstatus) |
| /api/v1/users.setActiveStatus | Set a user's active status. | [Link](/api/rest-api/methods/users/setactivestatus) |
| /api/v1/users.update | Update an existing user. | [Link](/api/rest-api/methods/users/update) |
| /api/v1/users.updateOwnBasicInfo | Update basic information of own user. | [Link](/api/rest-api/methods/users/updateownbasicinfo) |

## Channels

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/channels.addAll | Adds all of the users on the server to a channel. | [Link](/api/rest-api/methods/channels/addall) |
| /api/v1/channels.addLeader | Gives the role of Leader for a user in the current channel. | [Link](/api/rest-api/methods/channels/addleader) |
| /api/v1/channels.addOwner | Gives the role of owner for a user in the current channel. | [Link](/api/rest-api/methods/channels/addowner) |
| /api/v1/channels.anonymousread | Gets the messages in public channels to an anonymous user | [Link](/api/rest-api/methods/channels/anonymousread) |
| /api/v1/channels.archive | Archives a channel. | [Link](/api/rest-api/methods/channels/archive) |
| /api/v1/channels.cleanHistory | Cleans up a channel's history, requires special permission. | [Link](https://github.com/RocketChat/docs/tree/4f704d5da4a2d5bcfe2cd2d2591edd12f5d41cd6/developer-guides/rest-api/channels/cleanhistory/README.md) |
| /api/v1/channels.close | Removes a channel from a user's list of channels. | [Link](/api/rest-api/methods/channels/close) |
| /api/v1/channels.counters | Gets channel counters. | [Link](/api/rest-api/methods/channels/counters) |
| /api/v1/channels.create | Creates a new channel. | [Link](/api/rest-api/methods/channels/create) |
| /api/v1/channels.delete | Removes a channel. | [Link](/api/rest-api/methods/channels/delete) |
| /api/v1/channels.files | Gets a list of files from a channel. | [Link](/api/rest-api/methods/channels/files) |
| /api/v1/channels.getAllUserMentionsByChannel | Gets all the mentions of a channel. | [Link](/api/rest-api/methods/channels/getallusermentionsbychannel) |
| /api/v1/channels.getIntegrations | Gets the channel's integration. | [Link](/api/rest-api/methods/channels/getintegrations) |
| /api/v1/channels.history | Retrieves the messages from a channel. | [Link](/api/rest-api/methods/channels/history) |
| /api/v1/channels.info | Gets a channel's information. | [Link](/api/rest-api/methods/channels/info) |
| /api/v1/channels.invite | Adds a user to a channel. | [Link](/api/rest-api/methods/channels/invite) |
| /api/v1/channels.join | Joins yourself to a channel. | [Link](/api/rest-api/methods/channels/join) |
| /api/v1/channels.kick | Removes a user from a channel. | [Link](/api/rest-api/methods/channels/kick) |
| /api/v1/channels.leave | Removes the calling user from a channel. | [Link](/api/rest-api/methods/channels/leave) |
| /api/v1/channels.list | Retrieves all of the channels from the server. | [Link](/api/rest-api/methods/channels/list) |
| /api/v1/channels.list.joined | Gets only the channels the calling user has joined. | [Link](/api/rest-api/methods/channels/list-joined) |
| /api/v1/channels.members | Retrieves all channel users. | [Link](/api/rest-api/methods/channels/members) |
| /api/v1/channels.messages | Retrieves all channel messages. | [Link](/api/rest-api/methods/channels/messages) |
| /api/v1/channels.moderators | List all moderators of a channel. | [Link](/api/rest-api/methods/channels/moderators) |
| /api/v1/channels.online | List all online users of a channel. | [Link](/api/rest-api/methods/channels/online) |
| /api/v1/channels.open | Adds the channel back to the user's list of channels. | [Link](/api/rest-api/methods/channels/open) |
| /api/v1/channels.removeleader | Removes the role of Leader for a user in the current channel. | [Link](/api/rest-api/methods/channels/removeleader) |
| /api/v1/channels.rename | Changes a channel's name. | [Link](/api/rest-api/methods/channels/rename) |
| /api/v1/channels.roles | Gets the user's roles in the channel. | [Link](/api/rest-api/methods/channels/roles) |
| /api/v1/channels.setCustomFields | Sets a channel's custom fields. | [Link](/api/rest-api/methods/channels/setcustomfields) |
| /api/v1/channels.setAnnouncement | Sets a channel's announcement. | [Link](/api/rest-api/methods/channels/setannouncement) |
| /api/v1/channels.setDefault | Sets whether a channel is a default channel or not. | [Link](/api/rest-api/methods/channels/setdefault) |
| /api/v1/channels.setDescription | Sets a channel's description. | [Link](/api/rest-api/methods/channels/setdescription) |
| /api/v1/channels.setJoinCode | Sets the channel's code required to join it. | [Link](/api/rest-api/methods/channels/setjoincode) |
| /api/v1/channels.setPurpose | Sets a channel's description. | [Link](/api/rest-api/methods/channels/setpurpose) |
| /api/v1/channels.setReadOnly | Sets whether a channel is read only or not. | [Link](/api/rest-api/methods/channels/setreadonly) |
| /api/v1/channels.setTopic | Sets a channel's topic. | [Link](/api/rest-api/methods/channels/settopic) |
| /api/v1/channels.setType | Sets the type of room the channel should be. | [Link](/api/rest-api/methods/channels/settype) |
| /api/v1/channels.unarchive | Unarchives a channel. | [Link](/api/rest-api/methods/channels/unarchive) |
| /api/v1/channels.addOwner | Gives the role of owner for a user in the current channel. | [Link](/api/rest-api/methods/channels/addowner) |
| /api/v1/channels.removeOwner | Removes the role of owner from a user in the current channel. | [Link](/api/rest-api/methods/channels/removeowner) |

## Groups

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/groups.archive | Archives a private group. | [Link](/api/rest-api/methods/groups/archive) |
| /api/v1/groups.addLeader | Gives the role of Leader for a user in the current group. | [Link](/api/rest-api/methods/groups/addleader) |
| /api/v1/groups.close | Removes a private group from the list of groups. | [Link](/api/rest-api/methods/groups/close) |
| /api/v1/groups.counters | Gets group counters. | [Link](/api/rest-api/methods/groups/counters) |
| /api/v1/groups.create | Creates a new private group. | [Link](/api/rest-api/methods/groups/create) |
| /api/v1/groups.delete | Removes a private group. | [Link](/api/rest-api/methods/groups/delete) |
| /api/v1/groups.files | Gets a list of files from a private group. | [Link](/api/rest-api/methods/groups/files) |
| /api/v1/groups.history | Retrieves the messages from a private group. | [Link](/api/rest-api/methods/groups/history) |
| /api/v1/groups.info | Gets the information about a private group. | [Link](/api/rest-api/methods/groups/info) |
| /api/v1/groups.invite | Adds a user to the private group. | [Link](/api/rest-api/methods/groups/invite) |
| /api/v1/groups.kick | Removes a user from a private group. | [Link](/api/rest-api/methods/groups/kick) |
| /api/v1/groups.leave | Removes the calling user from the private group. | [Link](/api/rest-api/methods/groups/leave) |
| /api/v1/groups.list | List the private groups the caller is part of. | [Link](/api/rest-api/methods/groups/list) |
| /api/v1/groups.listAll | List all the private groups. | [Link](/api/rest-api/methods/groups/listall) |
| /api/v1/groups.members | Gets the users of participants of a private group. | [Link](/api/rest-api/methods/groups/members) |
| /api/v1/groups.messages | Retrieves all group messages. | [Link](/api/rest-api/methods/groups/messages) |
| /api/v1/groups.moderators | List all moderators of a group. | [Link](/api/rest-api/methods/groups/moderators) |
| /api/v1/groups.online | List all online users of a group. | [Link](https://github.com/RocketChat/docs/tree/aeb4dd8de5017b7cd9c9d9367a0e2155f911ba5a/api/rest-api/methods/groups/online.md) |
| /api/v1/groups.open | Adds the private group back to the list of groups. | [Link](/api/rest-api/methods/groups/open) |
| /api/v1/groups.removeLeader | Removes the role of Leader for a user in the current group. | [Link](/api/rest-api/methods/groups/removeleader) |
| /api/v1/groups.rename | Changes the name of the private group. | [Link](/api/rest-api/methods/groups/rename) |
| /api/v1/groups.roles | Gets the user's roles in the private group. | [Link](/api/rest-api/methods/groups/roles) |
| /api/v1/groups.setAnnouncement | Sets a group's announcement. | [Link](/api/rest-api/methods/groups/setannouncement) |
| /api/v1/groups.setCustomFields | Sets private group's custom fields. | [Link](/api/rest-api/methods/groups/setcustomfields) |
| /api/v1/groups.setDescription | Sets a private group's description. | [Link](/api/rest-api/methods/groups/setdescription) |
| /api/v1/groups.setPurpose | Sets a private group's description. | [Link](/api/rest-api/methods/groups/setpurpose) |
| /api/v1/groups.setReadOnly | Sets whether the room is read only or not. | [Link](/api/rest-api/methods/groups/setreadonly) |
| /api/v1/groups.setTopic | Sets a private group's topic. | [Link](/api/rest-api/methods/groups/settopic) |
| /api/v1/groups.setType | Sets the type of room this group will be. | [Link](/api/rest-api/methods/groups/settype) |
| /api/v1/groups.unarchive | Unarchives a private group. | [Link](/api/rest-api/methods/groups/unarchive) |
| /api/v1/groups.addOwner | Gives the role of owner for a user in the current group. | [Link](/api/rest-api/methods/groups/addowner) |
| /api/v1/groups.removeOwner | Removes the role of owner from a user in the current Group. | [Link](/api/rest-api/methods/groups/removeowner) |

## Chat

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/chat.delete | Deletes an existing chat message. | [Link](/api/rest-api/methods/chat/delete) |
| /api/v1/chat.followMessage | Follows an existing chat message. | [Link](/api/rest-api/methods/chat/followmessage) |
| /api/v1/chat.getDeletedMessages | Retrieves the deleted messages since specific date. | [Link](/api/rest-api/methods/chat/getdeletedmessages) |
| /api/v1/chat.getDiscussions | Retrieves the discussions messages of a room. | [Link](/api/rest-api/methods/chat/getdiscussions) |
| /api/v1/chat.getMentionedMessages | Retrieves mentioned messages. | [Link](/api/rest-api/methods/chat/getmentionedmessages) |
| /api/v1/chat.getMessage | Retrieves a single chat message. | [Link](/api/rest-api/methods/chat/getmessage) |
| /api/v1/chat.getMessageReadReceipts | Retrieves message read receipts. | [Link](/api/rest-api/methods/chat/getmessagereadreceipts) |
| /api/v1/chat.getPinnedMessages | Retrieve pinned messages from a room. | [Link](/api/rest-api/methods/chat/getpinnedmessages) |
| /api/v1/chat.getSnippetedMessages | Retrieves snippeted messages. | [Link](/api/rest-api/methods/chat/getsnippetedmessages) |
| /api/v1/chat.getSnippetedMessageById | Retrieves snippeted message by id. | [Link](/api/rest-api/methods/chat/getsnippetedmessagebyid) |
| /api/v1/chat.getStarredMessages | Retrieves starred messages. | [Link](/api/rest-api/methods/chat/getstarredmessages) |
| /api/v1/chat.getThreadMessages | Retrieves thread's messages. | [Link](/api/rest-api/methods/chat/getthreadmessages) |
| /api/v1/chat.getThreadsList | Retrieves channel's threads. | [Link](/api/rest-api/methods/chat/getthreadslist) |
| /api/v1/chat.ignoreUser | Ignores an user from a chat. | [Link](/api/rest-api/methods/chat/ignoreuser) |
| /api/v1/chat.pinMessage | Pins a chat message to the message's channel. | [Link](/api/rest-api/methods/chat/pinmessage) |
| /api/v1/chat.postMessage | Posts a new chat message. | [Link](/api/rest-api/methods/chat/postmessage) |
| /api/v1/chat.react | Sets/unsets the user's reaction to an existing chat message. | [Link](/api/rest-api/methods/chat/react) |
| /api/v1/chat.reportMessage | Reports a message. | [Link](/api/rest-api/methods/chat/reportmessage) |
| /api/v1/chat.search | Search for messages in a channel. | [Link](/api/rest-api/methods/chat/search) |
| /api/v1/chat.starMessage | Stars a chat message for the authenticated user. | [Link](/api/rest-api/methods/chat/starmessage) |
| /api/v1/chat.sendMessage | Send new chat message. | [Link](/api/rest-api/methods/chat/sendmessage) |
| /api/v1/chat.syncThreadMessages | Retrieves synced thread's messages. | [Link](/api/rest-api/methods/chat/syncthreadmessages) |
| /api/v1/chat.syncThreadsList | Retrieves thread's synced channel threads. | [Link](/api/rest-api/methods/chat/syncthreadslist) |
| /api/v1/chat.unfollowMessage | Unfollows an existing chat message. | [Link](/api/rest-api/methods/chat/unfollowmessage) |
| /api/v1/chat.unPinMessage | Removes the pinned status of the provided chat message. | [Link](/api/rest-api/methods/chat/unpinmessage) |
| /api/v1/chat.unStarMessage | Removes the star on the chat message for the authenticated user. | [Link](/api/rest-api/methods/chat/unstarmessage) |
| /api/v1/chat.update | Updates the text of the chat message. | [Link](/api/rest-api/methods/chat/update) |

## Custom Sounds

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/custom-sounds.list | Retrieves a list of custom sounds. | [Link](/api/rest-api/methods/custom-sounds/list) |

## IM

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/im.close | Removes a direct message from the list of direct messages. | [Link](/api/rest-api/methods/im/close) |
| /api/v1/im.counters | Gets counters of direct messages. | [Link](/api/rest-api/methods/im/counters) |
| /api/v1/im.create | Create a direct message session with another user. | [Link](/api/rest-api/methods/im/create) |
| /api/v1/im.history | Retrieves the messages from a direct message. | [Link](/api/rest-api/methods/im/history) |
| /api/v1/im.files | Retrieves a list of files from a direct message. | [Link](/api/rest-api/methods/im/files) |
| /api/v1/im.members | Retrieves the users of participants of a direct message. | [Link](/api/rest-api/methods/im/members) |
| /api/v1/im.messages | Retrieves the messages from specific direct message. | [Link](/api/rest-api/methods/im/messages) |
| /api/v1/im.messages.others | Retrieves the messages from any direct message in the server. | [Link](/api/rest-api/methods/im/messages-others) |
| /api/v1/im.list | List the direct messages the caller is part of. | [Link](/api/rest-api/methods/im/list) |
| /api/v1/im.list.everyone | List all direct message the caller in the server. | [Link](/api/rest-api/methods/im/list-everyone) |
| /api/v1/im.open | Adds the direct message back to the list of direct messages. | [Link](/api/rest-api/methods/im/open) |
| /api/v1/im.setTopic | Sets a direct message topic. | [Link](/api/rest-api/methods/im/settopic) |

## Integrations

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/integrations.create | Creates an integration. | [Link](/api/rest-api/methods/integration/create) |
| /api/v1/integrations.get | Gets an integration. | [Link](/api/rest-api/methods/integration/get) |
| /api/v1/integrations.history | Lists all history of the specified integration. | [Link](/api/rest-api/methods/integration/history) |
| /api/v1/integrations.list | Lists all of the integrations. | [Link](/api/rest-api/methods/integration/list) |
| /api/v1/integrations.remove | Removes an integration. | [Link](/api/rest-api/methods/integration/remove) |

## Invites

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/findOrCreateInvite | Created a new Invite or returns an existing one with the same parameters. | [Link](/api/rest-api/methods/invites/findorcreateinvite) |
| /api/v1/listInvites | Lists all of the invite tokens. | [Link](/api/rest-api/methods/invites/listinvites) |
| /api/v1/removeInvite | Removes an invite. | [Link](/api/rest-api/methods/invites/removeinvite) |
| /api/v1/useInviteToken | Report to the server that an invite token was used. | [Link](/api/rest-api/methods/invites/useinvitetoken) |
| /api/v1/validateInviteToken | Checks if an invite token is valid. | [Link](/api/rest-api/methods/invites/validateinvitetoken) |

## Livechat

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/livechat/inquiries.list | Retrieves a list of open inquiries. | [Link](/api/rest-api/methods/livechat/inquiries#inquiries-list) |
| /api/v1/livechat/inquiries.take | Take an open inquiry. | [Link](/api/rest-api/methods/livechat/inquiries#livechat-take-inquiry) |
| /api/v1/livechat/rooms | Retrieves a list of livechat rooms. | [Link](/api/rest-api/methods/livechat/rooms) |

## OAuthApps

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/oauth-apps.get | Retrieves an OAuth App by id or client id. | [Link](/api/rest-api/methods/oauthapps/get) |
| /api/v1/oauth-apps.list | Retrieves a list of OAuth Apps. | [Link](/api/rest-api/methods/oauthapps/list) |

## Permissions

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/permissions.listAll | Lists permissions on the server. | [Link](https://github.com/RocketChat/docs/tree/4f704d5da4a2d5bcfe2cd2d2591edd12f5d41cd6/developer-guides/rest-api/permissions/list-all/README.md) |
| /api/v1/permissions.update | Edits permissions on the server. | [Link](/api/rest-api/methods/permissions/update) |

## Roles

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/roles.create | Create a new role in the system. | [Link](/api/rest-api/methods/roles/create) |
| /api/v1/roles.list | Gets all the roles in the system. | [Link](/api/rest-api/methods/roles/list) |
| /api/v1/roles.sync | Gets all the roles in the system which are updated after a given date. | [Link](/api/rest-api/methods/roles/sync) |
| /api/v1/roles.addUserToRole | Assign a role to an user. | [Link](/api/rest-api/methods/roles/addusertorole) |
| /api/v1/roles.getUsersInRole | Gets the users that belongs to a role. | [Link](/api/rest-api/methods/roles/getusersinrole) |

## Push Token

|  |  |  |  |
| --- | --- | --- | --- |
| Url | Method | Short Description | Details Page |
| /api/v1/push.token | POST | Saves push token. | [Link](/api/rest-api/methods/push/push-token) |
| /api/v1/push.token | DELETE | Removes push token. | [Link](/api/rest-api/methods/push/deletepushtoken) |

## Rooms

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/rooms.adminRooms | Retrieve all rooms (requires special permission). | [Link](/api/rest-api/methods/rooms/adminrooms) |
| /api/v1/rooms.cleanHistory | Cleans up a room's history, requires special permission. | [Link](/api/rest-api/methods/rooms/cleanhistory) |
| /api/v1/rooms.createDiscussion | Creates a new discussion. | [Link](/api/rest-api/methods/rooms/creatediscussion) |
| /api/v1/rooms.favorite | Favorite/Unfavorite room. | [Link](/api/rest-api/methods/rooms/favorite) |
| /api/v1/rooms.get | Gets rooms. | [Link](/api/rest-api/methods/rooms/get) |
| /api/v1/rooms.getDiscussions | Gets room's discussions. | [Link](/api/rest-api/methods/rooms/getdiscussions) |
| /api/v1/rooms.info | Gets info from a room. | [Link](/api/rest-api/methods/rooms/info) |
| /api/v1/rooms.leave | Leaves a room. | [Link](/api/rest-api/methods/rooms/leave) |
| /api/v1/rooms.saveNotification | Sets the notifications settings of specific channel. | [Link](/api/rest-api/methods/rooms/savenotification) |
| /api/v1/rooms.upload/:rid | Upload a message with attached file. | [Link](/api/rest-api/methods/rooms/upload) |

## Command Methods

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/commands.get | Get specification of the slash command. | [Link](/api/rest-api/methods/commands/get) |
| /api/v1/commands.list | Lists all available slash commands. | [Link](/api/rest-api/methods/commands/list) |
| /api/v1/commands.run | Execute a slash command in the specified room. | [Link](/api/rest-api/methods/commands/run) |

## Custom User Status

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/custom-user-status.list | Lists all available custom user's status. | [Link](/api/rest-api/methods/custom-user-status/list) |

## Emoji Custom

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/emoji-custom.list | List the custom emojis available. | [Link](/api/rest-api/methods/emoji-custom/list) |
| /api/v1/emoji-custom.create | Create new custom emoji. | [Link](/api/rest-api/methods/emoji-custom/create) |
| /api/v1/emoji-custom.delete | Delete an existent custom emoji. | [Link](/api/rest-api/methods/emoji-custom/delete) |
| /api/v1/emoji-custom.update | Update an existent custom emoji. | [Link](/api/rest-api/methods/emoji-custom/update) |

## Settings

|  |  |  |  |
| --- | --- | --- | --- |
| Url | Method | Short Description | Details Page |
| /api/v1/settings | GET | Lists all private settings. | [Link](/api/rest-api/methods/settings/get) |
| /api/v1/settings.public | GET | Lists all public settings. | [Link](/api/rest-api/methods/settings/public) |
| /api/v1/settings.oauth | GET | Return list of all available oauth services. | [Link](/api/rest-api/methods/settings/oauth) |
| /api/v1/service.configurations | GET | Lists all service configurations. | [Link](/api/rest-api/methods/settings/service-configuration) |
| /api/v1/settings/:\_id | GET | Gets a setting. | [Link](/api/rest-api/methods/settings/get-by-id) |
| /api/v1/settings/:\_id | POST | Updates a setting. | [Link](/api/rest-api/methods/settings/update) |

## Subscriptions

|  |  |  |  |
| --- | --- | --- | --- |
| Url | Method | Short Description | Details Page |
| /api/v1/subscriptions.get | GET | Get all subscriptions. | [Link](/api/rest-api/methods/subscriptions/get) |
| /api/v1/subscriptions.getOne | GET | Get the subscription by room Id. | [Link](/api/rest-api/methods/subscriptions/getone) |
| /api/v1/subscriptions.read | POST | Mark a room as read. | [Link](/api/rest-api/methods/subscriptions/read) |
| /api/v1/subscriptions.unread | POST | Mark messages as unread. | [Link](/api/rest-api/methods/subscriptions/unread) |

## Video Conference

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/video-conference/jitsi.update-timeout | Updates the timeout of Jitsi video conference in a channel. | [Link](/api/rest-api/methods/video-conference/jitsi-update-timeout) |

## Webdav

|  |  |  |
| --- | --- | --- |
| Url | Short Description | Details Page |
| /api/v1/webdav.getMyAccounts | Retrieves the user's webdav accounts. | [Link](/api/rest-api/methods/webdav/getmyaccounts) |

# Language specific wrappers

## Java

* [rocket-chat-rest-client](https://github.com/baloise/rocket-chat-rest-client)

## PHP

* [rocket-chat-rest-client](https://github.com/Fab1en/rocket-chat-rest-client)

## Python

* [rocketchat\_API](https://github.com/jadolg/rocketchat_API)

## Ruby

* [rocketchat-ruby](https://github.com/abrom/rocketchat-ruby)

## Clojure

* [rocketchat-clojure](https://github.com/MalloZup/missile)

# **Livechat Widget API**

# Usage

Livechat Widget API code must be inserted after the Livechat Widget installation script and wrapped as a callback of RocketChat(); function.

You can call multiple Livechat Widget APIs on the same page.

## Methods

### Set custom field

To set a custom field for a visitor, you can use the following code:

RocketChat(function() {

this.setCustomField('fieldName1', 'Any value you want to store');

this.setCustomField('fieldName2', 'A value set just once', false); // you can pass false as the third parameter to not overwrite an already set value

});

### Set theme options

To change the online color of the Livechat widget, use the following code:

RocketChat(function() {

this.setTheme({

color: '#04436A', // widget title background color

fontColor: '#FFFFFF', // widget title font color

iconColor: '#1d74f5', // widget icon color

title: "Welcome to Rocket.Chat", // default widget title when the status of service is online

offlineTitle: "Service is offline", // default widget title when the status of service is online

});

});

### Assign chats to a specific department

To automatically assign a Livechat widget to a specific department (for example, to use a unique Livechat widget on more than one website), use the following code:

RocketChat(function() {

this.setDepartment('FILL HERE DEPARTMENT NAME - case sensitive');

});

### Set visitor token

To set an external token for a visitor, you can use the following code:

RocketChat(function() {

this.setGuestToken('FHwaLnp8fzjMupSAj');

});

### Set name field

To set the visitor name field, you can use the following code:

RocketChat(function() {

this.setGuestName('visitor name');

});

### Set email field

To set the visitor email field, you can use the following code:

RocketChat(function() {

this.setGuestEmail('sample@rocket.chat');

});

### Register visitor

To register the visitor without using the registration form, you can use the following code:

RocketChat(function() {

this.registerGuest({

token: 'FHwaLnp8fzjMupSAj', // The token field is not required. If it is not passed, a new token will be generated

name: 'visitor Name',

email: 'sample@rocket.chat',

department: 'my\_department', // The department field is not required,

customFields: [ // The customFields field is not required. If it is passed it needs to be an Array, where each item needs to be an object with key and value fields

{key: 'my\_custom\_field\_a', value: 'my\_custom\_field\_a\_value'},

{key: 'my\_custom\_field\_b', value: 'my\_custom\_field\_b\_value'}

]

});

});

### Set Language for Widget

There are number of language options, which you can choose from to set language for your widget. To check supported languages refer [here](https://github.com/RocketChat/Rocket.Chat.Livechat/tree/dev/src/i18n). To set language of widget use the following code.

RocketChat(function() {

this.setLanguage('af');

});

### Set a default Agent before starting a new conversation

The widget allows setting a specific agent before the conversation starts, to do this follow these steps:

RocketChat(function() {

this.setAgent({

\_id: 'h24yNtyoCmvp96wgt',

username: 'rocket.chat',

});

});

### Initialize the widget by configuring all available properties in just one call

The widget allows configuring all the settings in just one method, the following properties are acceptable:

RocketChat(function() {

this.initialize({

theme: {

color: '#04436A',

fontColor: '#FFFFFF',

iconColor: '#1d74f5',

title: "Welcome to Rocket.Chat",

offlineTitle: "Service is offline",

},

department: 'sales',

guestToken: 'FHwaLnp8fzjMupSAj',

language: 'en',

});

});

### Change widget visibility

You can either hide or show widget in your website. To hide widget use the following code:

RocketChat(function() {

this.hideWidget();

});

To show widget use the following code.

RocketChat(function() {

this.showWidget();

});

### Change widget window state

You can either open or close then widget in your website. To open widget(default state) use the following code:

RocketChat(function() {

this.maximizeWidget();

});

To close the widget use the following code.

RocketChat(function() {

this.minimizeWidget();

});

## Events

### onChatMaximized

Fired when the chat widget is maximized.

RocketChat(function() {

this.onChatMaximized(function() {

// do whatever you want

console.log('chat widget maximized');

});

});

### onChatMinimized

Fired when the chat widget is minimized.

RocketChat(function() {

this.onChatMinimized(function() {

// do whatever you want

console.log('chat widget minimized');

});

});

### onChatStarted

Fired when the chat is started (the first message was sent).

RocketChat(function() {

this.onChatStarted(function() {

// do whatever you want

console.log('chat started');

});

});

### onChatEnded

Fired when the chat is ended either by the agent or the visitor.

RocketChat(function() {

this.onChatEnded(function() {

// do whatever you want

console.log('chat ended');

});

});

### onPrechatFormSubmit

Fired when the pre-chat form is submitted.

RocketChat(function() {

this.onPrechatFormSubmit(function(data) {

// data is an object containing the following fields: name, email and deparment (the department \_id)

​

// do whatever you want

console.log('pre-chat form submitted');

});

});

### onOfflineFormSubmit

Fired when the offline form is submitted.

RocketChat(function() {

this.onOfflineFormSubmit(function(data) {

// data is an object containing the following fields: name, email and message

​

// do whatever you want

console.log('offline form submitted');

});

});

### onWidgetHidden

Fired when widget is hidden.

RocketChat(function() {

this.onWidgetHidden(function(data) {

// do whatever you want

console.log('chat widget hidden');

});

});

### onAssignAgent

Fired when an agent is assigned to the chat.

RocketChat(function() {

this.onAssignAgent(function(data) {

// data is an object containing the following fields: name, username and status

​

// do whatever you want

console.log('Agent assigned');

});

});

### onWidgetShown

Fired when widget is shown.

RocketChat(function() {

this.onWidgetShown(function(data) {

// do whatever you want

console.log('chat widget shown');

});

});

### onAgentStatusChange

Fired when the status of the current agent changes.

RocketChat(function() {

this.onAgentStatusChange(function(data) {

// data is an object containing the following fields: name, username and status

​

// do whatever you want

console.log('The status of the agent has changed');

});

});

### onServiceOffline

Fired when a visitor tries to start a new conversation and the Livechat service is offline.

RocketChat(function() {

this.onServiceOffline(function(data) {

// do whatever you want

console.log('The Livechat service is offline');

});

});

## Add Rocket.chat into your website:

# **Connect to a server**

Use a web browser, the Rocket.Chat desktop/mobile application, or your choice of [client applications](https://rocket.chat/download) to connect to a Rocket.Chat server.

# Use a web browser

To connect to a Rocket.Chat server through a web browser, enter the desired server address in the browser's address bar such as https://open.rocket.chat.

After entering a valid Rocket.Chat server address, the [registration](https://docs.rocket.chat/guides/user-guides/registration) and [log in](https://docs.rocket.chat/guides/user-guides/login) page appears. Enter your credentials to log in or register to become a member of that server.

# Use the desktop app

To connect to a Rocket.Chat server using the desktop app:

1. ​[Download](https://rocket.chat/download) the app specific to your operating system.
2. Install and open the app.
3. The server connect screen appears. To connect to a server, enter a valid server address and click **Connect**.

After entering a valid Rocket.Chat server address, the [registration](https://docs.rocket.chat/guides/user-guides/registration) and [log in](https://docs.rocket.chat/guides/user-guides/login) page appears. Enter your credentials to log in or register to become a member of that server.

## Connect to more than one server

When using a Rocket.Chat desktop application you can connect to multiple Rocket.Chat servers.

To add a new server, click the **plus** icon below the server list on the left sidebar and follow the [instructions](https://docs.rocket.chat/guides/user-guides/connecting-to-a-server) to connect to a Rocket.Chat server.

Use the mobile app

To connect to a Rocket.Chat server using a mobile client app [download](https://rocket.chat/download) the app for your mobile operating system. To find the app, search for "Rocket.Chat" in either Google Play or App Store. Download the application and install.

Open the app. If this is your first time using the mobile app, the server connect screen appears. Press **Connect to a Server**, enter a valid server address and then click **Connect**.

After entering a valid Rocket.Chat server address, the [registration](https://docs.rocket.chat/guides/user-guides/registration) and [log in](https://docs.rocket.chat/guides/user-guides/login) page appears. Enter your credentials to log in or register to become a member of that server.

# **Logging into a server**

To login to a server [register](https://docs.rocket.chat/guides/user-guides/registration) an account manually or with a pre-configured OAuth service.

# Manual Log in

To manually log in, you must have used the [manual registration method](https://docs.rocket.chat/guides/user-guides/registration) or have set a password in your account settings.

You can log in using either your username or email and password.

# OAuth Log in

Your server's system administrator enables OAuth login. If an OAuth service is enabled, click the icon for that service (i.e., GitHub or Google), and then follow the on-screen prompts to log in.

# **Iframe Integration**

# Use your own login page to login users in Rocket.Chat

If you want to authenticate users using your own login page in place of the Rocket.Chat's login page via the iframe integration, go to Administration > Accounts > Iframe and enable it:

* ​[How to use iframe integration for authentication](https://docs.rocket.chat/guides/developer/iframe-integration/authentication)​

Use Rocket.Chat in your site/app inside an iframe

If you need listen to events, go to Administration > General > Iframe Integration and enable send (events) or receive (commands), depending on your needs:

* ​[How to use iframe integration events](https://docs.rocket.chat/guides/developer/iframe-integration/events)​
* ​[How to use iframe integration commands](https://docs.rocket.chat/guides/developer/iframe-integration/commands)​
* ​[Using embedded layout mode](https://docs.rocket.chat/guides/developer/embedded-layout)​

# **What is iframe auth?**

With iframe auth you can use your own authentication page/API to log in users on Rocket.Chat.

When enabled Rocket.Chat first do an XMLHttpRequest to the iFrame API URL trying to see if the user is already logged in at the third party website. If that doesn't succeed then Rocket.Chat will present the Iframe URL within an iframe, so the user logs in on the third party website which means he is authenticated on Rocket.Chat as well.

We have developed an example app written in NodeJS in order to help you understanding this authentication flow. Please take a look at [iFrame Auth Example](https://github.com/RocketChat/iframe-auth-example)​

# Configuring

## API URL and API Method

Configure how Rocket.Chat will call the third party system to either login or to verify if the user is already logged in, by setting API URL and API Method fields.

API URL refers to endpoint on the third-party system that will check if the user is already logged in to that system. The API Method is used to select the submission method Rocket.Chat will use to submit information to the API URL (for instance using POST).

If the user has already logged into the third-party system, the API URL should communicate to Rocket.Chat and return a JSON object containing either a token or loginToken property, otherwise (if the user is not already logged in) the API URL should return an empty body with status 401.

The choice of which property API URL will return depends on how the third-party system decides to interface back with Rocket.Chat, as described in one of the two ways below:

### Using Rocket.Chat API

If you have the user's password stored (or it is the same between your third party system and Rocket.Chat), you can use [Rocket.Chat's REST APIs](https://docs.rocket.chat/developer-guides/rest-api/authentication/login/) to log in the user, this way you will get an authToken back from Rocket.Chat that should be returned as loginToken by your endpoint.

At this point, if the user does not have a Rocket.Chat account yet, you can either use Rocket.Chat API to [create an user](https://docs.rocket.chat/developer-guides/rest-api/users/create/) using a admin account or [register him](https://docs.rocket.chat/developer-guides/rest-api/users/register/).

After you log the user in, you should return a payload like the following:

{

"loginToken": "already-saved-or-returned-login-token"

}

### Managing MongoDB directly

In the case you have access to Rocket.Chat's database, you can connect there directly and manage the user record by yourself. This might be useful if you have MongoDB on your stack already and don't want to learn Rocket.Chat's API.

To do so the endpoint should connect on Rocket.Chat's MongoDB database and make sure the generated-token is saved on users collection on the corresponding user record. The generated-token should be saved on the field path services.iframe.token. This is how the user record should look like:

{

"\_id": "MZiFvWAf96876875u",

"createdAt": new Date(1432252673528),

"services": {

"iframe": {

"token": "generated-token"

}

},

"emails": [

{

"address": "useremail@gmail.com",

"verified": true

}

],

"name": "John Doe",

"username": "john.doe",

"active": true,

"statusDefault": "online",

"roles": [

"user"

],

"type": "user"

}

On this case, the response should be:

{

"token": "generated-token"

}

## IFrame URL

The URL of the page you want to show as the login page of your Rocket.Chat instance (this page can be created in any programming language and/or web framework).

The login page will then communicate back to Rocket.Chat using postMessage API.

After user logs in, you have to authenticate him on Rocket.Chat side, pretty much the same as you did before on API URL endpoint, but now you should return a JavaScript code that will be rendered within the iframe, depending how you logged in the user:

* If have used Rocket.Chat's APIs to log in the user or already have user's token saved in your end, return:

<script>

window.parent.postMessage({

event: 'login-with-token',

loginToken: 'your-token'

}, 'http://your.rocket.chat.url');

</script>

* If you have saved user's token connecting directly to Rocket.Chat's database on the user's field services.iframe.token:

<script>

window.parent.postMessage({

event: 'try-iframe-login'

}, 'http://your.rocket.chat.url');

</script>

## Using OAuth configured on Rocket.Chat's end

If you have OAuth services configured on Rocket.Chat, you can trigger them from within your login page as well.

To implement this authentication, after triggering the OAuth authentication you will receive a postMessage back from Rocket.Chat with user's credentials response from OAuth service. You need to manage the user creation/authentication on Rocket.Chat's database by yourself, the same as described earlier.

### Facebook

window.parent.postMessage({

event: 'call-facebook-login',

permissions: ['email']

}, 'http://your.rocket.chat.url');

The reply will be a postMessage back to your page with:

{

event: 'facebook-login-success',

response: {

// authResponse: Object

// accessToken: "a7s6d8a76s8d7..."

// expiresIn: "5172793"

// secret: "..."

// session\_key: true

// sig: "..."

// userID: "675676576"

// status: "connected"

}

}

Or an error

{

event: 'facebook-login-error',

error: error,

response: response

}

### Google

window.parent.postMessage({

event: 'call-google-login',

// scopes:

// webClientId:

}, 'http://your.rocket.chat.url');

The reply will be a postMessage back to your page with:

{

event: 'google-login-success',

response: {

// "email": "rodrigoknascimento@gmail.com",

// "userId": "1082039180239",

// "displayName": "Rodrigo Nascimento",

// "gender": "male",

// "imageUrl": "https://lh5.googleusercontent.com/-shUpniJA480/AAAAAAAAAAI/AAAAAAAAAqY/\_B8oyS8yBw0/photo.jpg?sz=50",

// "givenName": "Rodrigo",

// "familyName": "Nascimento",

// "ageRangeMin": 21,

// "oauthToken": "123198273kajhsdh1892h"

}

}

Or an error

{

event: 'google-login-error',

error: error

}

### Twitter

window.parent.postMessage({

event: 'call-twitter-login'

}, 'http://your.rocket.chat.url');

The reply will be a postMessage back to your page with:

{

event: 'twitter-login-success',

response: {

// "userName": "orodrigok",

// "userId": 293123,

// "secret": "asdua09sud",

// "token": "2jh3k1j2h3"

}

}

Or an error

{

event: 'twitter-login-error',

error: error

}

# How to login in Rocket.Chat with default account system while in development

When you activate the IFrame auth you will not be able to access Rocket.Chat's default login page, however if still need/want to use your Rocket.Chat's credentials to log in, you can do that by opening the browser's Developer Console and executing the following code:

Meteor.loginWithPassword('username-or-email', 'your-password');