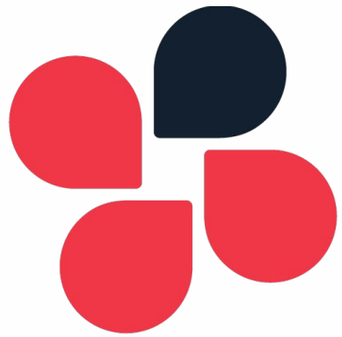

Chatwork

Chatwork API Documentation

Changelog

Date	Changes
2022/10/6	Updated APIs as announced here to align with the service content change for Basic plan.
2024/8/29	Updated APIs as announced here to align with the service content change for Free plan.
2025/4/3	Updated APIs as announced here to align with the removing certain query parameters.

What’s Chatwork API?

Chatwork API is an API provided for developers to programmatically interact with Chatwork's services. It enables you to integrate Chatwork’s features into your own web application and business system. Chatwork API can be used to make a web server notify the administrator when an error occurs, or make a project management tool send out messages to other Chatwork users when tasks status are updated.

Before You Start

If you are using Business or Enterprise plan, you need to have a permission from admin user to use APIs.
Please submit your request from the link below.

Submit Your Request

Chatwork API Endpoints

Chatwork API Endpoints

Chatwork API is designed according to the REST architectural style. In Chatwork API, a unique URI is given to each of the API resource. These are called API endpoints. You can interact with the Chatwork API by passing parameters through HTTP request methods such as GET, POST, PUT, and DELETE.

The base URI for the API endpoint is https://api.chatwork.com/v2.
You will need to add the unique endpoint path to the base URI path to use the desired API method (for example, to use “/me” you will write “https://api.chatwork.com/v2/me”).

In the future, when the API version is updated, "/v2" part of the path will be modified.
You will be given a certain transition time to accommodate the change when an API version is updated.

HTTP Request

Chatwork API requires that connection to the API endpoint is made with **https** (secure protocol). Using http will result in a connection error.

As it was discussed earlier in the Authentication section, every HTTP request must include the API token in its header field. The key name used to set the value of the API token in the HTTP header field is "`x-chatworktoken`".

[IMPORTANT] Since API tokens do not have expiration date, it is important to keep the API token secure. Be sure to set the API token in the HTTP header field and NOT in the URL query string.

HTTP Response

The HTTP response you get back from the Chatwork API will be in JSON format. The response body in JSON format will not include any information to indicate whether or not the request was processed successfully. To find out if you have a successful response, you can check the HTTP status code in the response header.

The output below shows the result from executing "GET /my/status" which returns you information about how many unread messages you have and more.

HTTP Response Header

```
HTTP/2 200
content-type: application/json
```

HTTP Response Body

```
{
  "unread_room_num": 2,
  "mention_room_num": 1,
  "mytask_room_num": 3,
  "unread_num": 12,
  "mention_num": 1,
  "mytask_num": 8
}
```

HTTP Response

If an error occurs, the response header will contain the corresponding HTTP status code and the response body will contain the error message in JSON format.
Below is a case where the API token was not valid.

HTTP Response Header

```
HTTP/2 401
content-type: application/json
```

HTTP Response Body

```
{
  "errors": ["Invalid API token"]
}
```

Error message will be returned in array format under key name "`errors`".

API Usage Limits

Number of API requests you can make in 5 minutes is limited to 300 times (This limit could possibly change in the future). Number of remaining API calls and the reset time can be obtained from the response header.

Example Response Header

```
HTTP/2 200
content-type: application/json
x-ratelimit-limit: 300
x-ratelimit-remaining: 244
x-ratelimit-reset: 1390941626
```

Each field has the following meaning:

`x-ratelimit-limit`: Maximum number of calls you can make in the time frame
`x-ratelimit-remaining`: Remaining number of calls you can make
`x-ratelimit-reset`: Time at which the limit will next be reset (Unix time)

If you exceed the limit, the API will return response with status code `429`.

Example Response Header

```
HTTP/2 429
content-type: application/json
x-ratelimit-limit: 300
x-ratelimit-remaining: 0
x-ratelimit-reset: 1390941626
```

Messages API Limitations for Free Plan

In the Free Plan, the past 40 days messages are accessible. Therefore, some or all of the messages may not be available for the following APIs.

- GET: /rooms/{room_id}/messages
- GET: /rooms/{room_id}/messages/{message_id}

The following response headers will be added to determine if the limitation is applied or not.

`chatwork-message-limitation`

- The only valid value for this header is `true`.
- If the limitation is not applied, then this header will be omitted.

`chatwork-message-limitation-summary`

- Summary of the limitation will be set.

Chatwork API Endpoints

- /me
  - GET: /me
- /my
  - GET: /my/status
  - GET: /my/tasks
- /contacts
  - GET: /contacts
- /rooms
  - GET: /rooms
  - POST: /rooms
  - GET: /rooms/{room_id}
  - PUT: /rooms/{room_id}
  - DELETE: /rooms/{room_id}
  - GET: /rooms/{room_id}/members
  - PUT: /rooms/{room_id}/members
  - GET: /rooms/{room_id}/messages
  - POST: /rooms/{room_id}/messages
  - GET: /rooms/{room_id}/messages/{message_id}
  - GET: /rooms/{room_id}/tasks
  - POST: /rooms/{room_id}/tasks
  - GET: /rooms/{room_id}/tasks/{task_id}
  - GET: /rooms/{room_id}/files
  - GET: /rooms/{room_id}/files/{file_id}

Endpoints: /me

Used to access your account information.

[GET] /me

Get your account information.

```
curl -X GET -H "x-chatworktoken: Your API token" "https://api.chatwork.com/v2/me”
```

Response:


```
{
  "account_id": 123,
  "room_id": 322,
  "name": "John Smith",
  "chatwork_id": "tarochatworkid",
  "organization_id": 101,
  "organization_name": "Hello Company",
  "department": "Marketing",
  "title": "CMO",
  "url": "http://mycompany.com",
  "introduction": "Self Introduction",
  "mail": "taro@example.com",
  "tel_organization": "XXX-XXXX-XXXX",
  "tel_extension": "YYY-YYYY-YYYY",
  "tel_mobile": "ZZZ-ZZZZ-ZZZZ",
  "skype": "myskype_id",
  "facebook": "myfacebook_id",
  "twitter": "mytwitter_id",
  "avatar_image_url": "https://example.com/abc.png"
}
```

Endpoints: /my

Used to access your data on the account.

[GET] /my/status

Get the number of: unread messages, unread To messages, and unfinished tasks.

```
curl -X GET -H "x-chatworktoken: Your API token" "https://api.chatwork.com/v2/my/status"
```

Response:


```
{
  "unread_room_num": 2,
  "mention_room_num": 1,
  "mytask_room_num": 3,
  "unread_num": 12,
  "mention_num": 1,
  "mytask_num": 8
}
```

Endpoints: /my

[GET] /my/tasks

Get the list of all unfinished tasks (*This method returns up to 100 entries. We are planning to implement pagination to support larger number of data retrieval).

```
curl -X GET -H "x-chatworktoken: Your API token"
"https://api.chatwork.com/v2/my/tasks?assigned_by_account_id=78&status=done"
```

Response:


```
[
  {
    "task_id": 3,
    "room": {
      "room_id": 5,
      "name": "Group Chat Name",
      "icon_path": "https://example.com/ico_group.png"
    },
    "assigned_by_account": {
      "account_id": 78,
      "name": "Anna",
      "avatar_image_url": "https://example.com/def.png"
    },
    "message_id": "13",
    "body": "buy milk",
    "limit_time": 1384354799,
    "status": "open"
  }
]
```


[Parameter]

assigned_by_account_id [integer]
Account ID of the person who assigned task.

Example input	78

status [string] [enum]
Task status

enum values:

`open, done`

Example input	done

Endpoints: /contacts

Used to access the list of your contacts.

[GET] /contacts

Get the list of your contacts.

```
curl -X GET -H "x-chatworktoken: Your API token" "https://api.chatwork.com/v2/contacts”
```

Response:


```
[
  {
    "account_id": 123,
    "room_id": 322,
    "name": "John Smith",
    "chatwork_id": "tarochatworkid",
    "organization_id": 101,
    "organization_name": "Hello Company",
    "department": "Marketing",
    "avatar_image_url": "https://example.com/abc.png"
  }
]
```

Endpoints: /rooms

Used to access information such as messages, members, files, and tasks associated to a specific conversation. The conversation can be Group chat, Direct chat, or My chat.

[GET] /rooms

Get the list of all chats on your account.

```
curl -X GET -H "x-chatworktoken: Your API token" "https://api.chatwork.com/v2/rooms"
```

Response:


```
[
  {
    "room_id": 123,
    "name": "Group Chat Name",
    "type": "group",
    "role": "admin",
    "sticky": false,
    "unread_num": 10,
    "mention_num": 1,
    "mytask_num": 0,
    "message_num": 122,
    "file_num": 10,
    "task_num": 17,
    "icon_path": "https://example.com/ico_group.png",
    "last_update_time": 1298905200
  }
]
```

Endpoints: /rooms

[POST] /rooms

Create a new group chat.

```
curl -X POST -H "x-chatworktoken: Your API token" -d
"description=group+chat+description&icon_preset=meeting&members_admin_ids=123%2C542%2C1001&membe
rs_member_ids=21%2C344&members_readonly_ids=15%2C103&name=Website+renewal+project"
"https://api.chatwork.com/v2/rooms”
```

Response:

```
{
  "room_id": 1234
}
```

Endpoints: /rooms

[POST] /rooms

[Parameter]

`description` [string]
Chat Description
Description of the group chat.

Example input	`group chat description`

`icon_preset` [string][enum]
Icon Type
Type of the group chat icon.

enum values:

```
group, check, document, meeting, event, project, business, study,
security, star, idea, heart, magcup, beer, music, sports, travel
```

Example input	`meeting`

`members_admin_ids` [integer] [list] [required]
Group Chat Administrators
List of user IDs who will be given administrator permission for the group chat. At least one user must be specified as an administrator.

Example input	`123,542,1001`

`members_member_ids` [integer] [list]
Group Chat Members
List of user IDs who will be given member permission for the group chat.

Example input	`21,344`

`members_readonly_ids` [integer] [list]
Group Chat Read-only Users
List of user IDs who will be given read-only permission for the group chat.
*Use commas to separate each entry in the list

Example input	`15,103`

`members_admin_ids` [string] [required]
Group Chat Name
Title of the group chat.

Example input	`Website renewal project`

Endpoints: /rooms

[GET] /rooms/{room_id}

Get chat name, icon, and Type (`my`, `direct`, or `group`).

```
curl -X GET -H "x-chatworktoken: Your API token" "https://api.chatwork.com/v2/rooms/{room_id}"
```

Response:


```
{
  "room_id": 123,
  "name": "Group Chat Name",
  "type": "group",
  "role": "admin",
  "sticky": false,
  "unread_num": 10,
  "mention_num": 1,
  "mytask_num": 0,
  "message_num": 122,
  "file_num": 10,
  "task_num": 17,
  "icon_path": "https://example.com/ico_group.png",
  "last_update_time": 1298905200,
  "description": "room description text"
}
```

Endpoints: /rooms

[PUT] /rooms/{room_id}

Change the title and icon type of the specified chat.

```
curl -X PUT -H "x-chatworktoken: Your API token" -d
"description=group+chat+description&icon_preset=meeting&name=Website+renewal+project"
"https://api.chatwork.com/v2/rooms/{room_id}”
```

Response:

```
{
  "room_id": 1234
}
```

[Parameter]

`description` [string]
Chat Description
Description of the group chat.

Example input	`group chat description`

`name` [string]
Group Chat Name
Title of the group chat.

Example input	`Website renewal project`

`icon_preset` [string] [enum]
Icon Type
Type of the group chat icon.

enum values:

```
group, check, document, meeting, event, project, business, study,
security, star, idea, heart, magcup, beer, music, sports, travel
```

Example input	`meeting`

Endpoints: /rooms

[DELETE] /rooms/{room_id}

Leave/Delete a group chat.
When a user leaves the group chat, all the user's tasks and files uploaded by the user will be deleted.
When a group chat is deleted, all the messages, tasks, and files associated with the group chat will also be deleted.
Please be aware that deleted contents cannot be restored.

```
curl -X DELETE -H "x-chatworktoken: Your API token" -d "action_type=leave"
"https://api.chatwork.com/v2/rooms/{room_id}”
```

Response:

204

[Parameter]

`action_type` [string] [enum] [required]
Type of action

enum values:

```
leave, delete
```

Example input	`leave`

Endpoints: /rooms

[GET] /rooms/{room_id}/members

Get the list of all chat members associated with the specified chat .

```
curl -X GET -H "x-chatworktoken: Your API token"
"https://api.chatwork.com/v2/rooms/{room_id}/members”
```

Response:

```
[
  {
    "account_id": 123,
    "role": "member",
    "name": "John Smith",
    "chatwork_id": "tarochatworkid",
    "organization_id": 101,
    "organization_name": "Hello Company",
    "department": "Marketing",
    "avatar_image_url": "https://example.com/abc.png"
  }
]
```

Endpoints: /rooms

[PUT] /rooms/{room_id}/members

Change associated members of group chat at once.

```
curl -X PUT -H "x-chatworktoken: Your API token" -d
"members_admin_ids=123%2C542%2C1001&members_member_ids=21%2C344&members_readonly_ids=15%2C103"
"https://api.chatwork.com/v2/rooms/{room_id}/members”
```

Response:

```
{
  "admin": [123, 542, 1001],
  "member": [10, 103],
  "readonly": [6, 11]
}
```

[Parameter]

members_admin_ids [integer] [list] [required]
Group Chat Administrators
List of user IDs who will be given administrator permission for the group chat. At least one user must be specified as an administrator.

Example input	123,542,1001

members_member_ids [integer] [list]
Group Chat Members
List of user IDs who will be given member permission for the group chat.

Example input	21,344

members_readonly_ids [integer] [list]
Group Chat Read-only Users
List of user IDs who will be given read-only permission for the group chat.

Example input	15,103

Endpoints: /rooms

[GET] /rooms/{room_id}/messages

Get all messages associated with the specified chat (returns up to 100 entries).
If the parameter is not set, it returns the next 100 entries from previous call.

```
curl -X GET -H "x-chatworktoken: Your API token" "https://api.chatwork.com/v2/rooms/{room_id}/messages”
```

Response:


```
[
  {
    "message_id": "5",
    "account": {
      "account_id": 123,
      "name": "Bob",
      "avatar_image_url": "https://example.com/ico_avatar.png"
    },
    "body": "Hello Chatwork!",
    "send_time": 1384242850,
    "update_time": 0
  }
]
```


[Parameter]

`force` [boolean]
Flag which forces to get 100 newest entries regardless of previous calls.
※Set to 1 to start from the beginning regardless of previous calls (`0` is the default value).

Example input	`0`

Please also refer to [Messages API Limitations for Free Plan].

Endpoints: /rooms

[POST] /rooms/{room_id}/messages

Add new message to the chat.

```
curl -X POST -H "x-chatworktoken: Your API token" -d "body=Hello+Chatwork%21&self_unread=0"
"https://api.chatwork.com/v2/rooms/{room_id}/messages”
```

Response:

```
{
  "message_id": "1234"
}
```

[Parameter]

body [string] [required]
message body

Example input	`Hello Chatwork!`

`self_unread` [boolean]
Make the messages you posted unread.
※By setting this as `1`, the messages you posted is turned to unread (`0` is the default value: read).

Example input	`0`

Endpoints: /rooms

[GET] /rooms/{room_id}/messages/{message_id}

Get information about the specified message.

```
curl -X GET -H "x-chatworktoken: Your API token"
"https://api.chatwork.com/v2/rooms/{room_id}/messages/{message_id}”
```

Response:


```
{
  "message_id": "5",
  "account": {
    "account_id": 123,
    "name": "Bob",
    "avatar_image_url": "https://example.com/ico_avatar.png"
  },
  "body": "Hello Chatwork!",
  "send_time": 1384242850,
  "update_time": 0
}
```


Please also refer to [Messages API Limitations for Free Plan](#).

Endpoints: /rooms

[GET] /rooms/{room_id}/tasks

Get the list of tasks associated with the specified chat (*This method returns up to 100 entries. We are planning to implement pagination to support larger number of data retrieval).

```
curl -X GET -H "x-chatworktoken: Your API token"
"https://api.chatwork.com/v2/rooms/{room_id}/tasks?account_id=101&assigned_by_account_id=78&status=done”
```

Response:


```
[
  {
    "task_id": 3,
    "account": {
      "account_id": 101,
      "name": "Bob",
      "avatar_image_url": "https://example.com/abc.png"
    },
    "assigned_by_account": {
      "account_id": 456,
      "name": "Anna",
      "avatar_image_url": "https://example.com/def.png"
    },
    "message_id": "13",
    "body": "buy milk",
    "limit_time": 1384354799,
    "status": "open"
  }
]
```


[Parameter]

account_id [integer]
Account ID of the person responsible to complete the task.

Example input	101

assigned_by_account_id [integer]
Group Chat Members
Account ID of the person who assigned the task.

Example input	78

status [integer] [enum]
Status of the task.

enum values:

```
open, done
```

Example input	done

Endpoints: /rooms

[POST] /rooms/{room_id}/tasks

Add a new task to the chat.

```
curl -X POST -H "x-chatworktoken: Your API token" -d "body=Buy+milk&limit=1385996399&to_ids=1%2C3%2C6"
"https://api.chatwork.com/v2/rooms/{room_id}/tasks"
```

Response:

```
{
  "task_ids": [123,124]
}
```

[Parameter]

body [string] [required]
Task description.

Example input	Buy milk

limit [integer]
When the task is due.
*Use Unix time as input

Example input	1385996399

to_ids [integer] [list] [required]
Account ID of the person/people responsible to complete the task
If multiple, IDs must be separated by comma.

Example input	136

Endpoints: /rooms

[GET] /rooms/{room_id}/tasks/{task_id}

Get information about the specified task.

```
curl -X GET -H "x-chatworktoken:  (Your API Token)"
"https://api.chatwork.com/v2/rooms/{room_id}/tasks/{task_id}”
```

Response:


```
{
  "task_id": 3,
  "account": {
    "account_id": 123,
    "name": "Bob",
    "avatar_image_url": "https://example.com/abc.png"
  },
  "assigned_by_account": {
    "account_id": 456,
    "name": "Anna",
    "avatar_image_url": "https://example.com/def.png"
  },
  "message_id": "13",
  "body": "buy milk",
  "limit_time": 1384354799,
  "status": "open"
}
```

Endpoints: /rooms

[GET] /rooms/{room_id}/files

Get the list of files associated with the specified chat
(*This method returns up to 100 entries. We are planning to implement pagination to support larger number of data retrieval).

```
curl -X GET -H "x-chatworktoken: (Your API Token)"
"https://api.chatwork.com/v2/rooms/{room_id}/files?account_id=101"
```

Response:


```
[
  {
    "file_id": 3,
    "account": {
      "account_id": 101,
      "name": "Bob",
      "avatar_image_url": "https://example.com/ico_avatar.png"
    },
    "message_id": "22",
    "filename": "README.md",
    "filesize": 2232,
    "upload_time": 1384414750
  }
]
```


[Parameter]

`account_id` [integer]
Account ID of the person who uploaded the file.

Example input	101

Endpoints: /rooms

[GET] /rooms/{room_id}/files/{file_id}

Get information about the specified file.

```
curl -X GET -H "x-chatworktoken: (Your API Token)"
"https://api.chatwork.com/v2/rooms/{room_id}/files/{file_id}?create_download_url=1”
```

Response:


```
{
  "file_id":3,
  "account": {
    "account_id":123,
    "name":"Bob",
    "avatar_image_url": "https://example.com/ico_avatar.png"
  },
  "message_id": "22",
  "filename": "README.md",
  "filesize": 2232,
  "upload_time": 1384414750
}
```


[Parameter]

`create_download_url` [boolean]
Whether or not to create a download link.
If set to true, download like will be created for 30 seconds.

Example input	1

Endpoints: /incoming_requests

You can access contact approval requests you received.

[GET] /incoming_requests

You can get the list of contact approval request you received
(*This method returns up to 100 entries. We are planning to implement pagination to support larger number of data retrieval).

```
curl -X GET -H "x-chatworktoken: (Your API Token)"
"https://api.chatwork.com/v2/incoming_requests"
```

Response:

```
[
  {
    "request_id": 123,
    "account_id": 363,
    "message": "hogehoge",
    "name": "John Smith",
    "chatwork_id": "tarochatworkid",
    "organization_id": 101,
    "organization_name": "Hello Company",
    "department": "Marketing",
    "avatar_image_url": "https://example.com/abc.png"
  }
]
```

Endpoints: /incoming_requests

[PUT] /incoming_requests/{request_id}

You can approve a contact approval request you received.

```
curl -X PUT -H "x-chatworktoken: (Your API Token)"
"https://api.chatwork.com/v2/incoming_requests/{request_id}"
```

Response:


```
{
  "account_id": 363,
  "room_id": 1234,
  "name": "John Smith",
  "chatwork_id": "tarochatworkid",
  "organization_id": 101,
  "organization_name": "Hello Company",
  "department": "Marketing",
  "avatar_image_url": "https://example.com/abc.png"
}
```

Endpoints: /incoming_requests

[DELETE] DELETE/incoming_requests/{request_id}

You can decline a contact approval request you received.

```
curl -X DELETE -H "x-chatworktoken: (Your API Token)"
"https://api.chatwork.com/v2/incoming_requests/{request_id}"
```

Response:

204

OAuth

Authorization flow

Of the authorization flows (authorization grants) described in RFC6749, you can only use the **Authorization Code Grant** type.

Features

All of the current Chatwork APIs are available.

How to use a Chatwork API from a client program

1. Client registration

You can register clients from the OAuth client creation screen.

Register the following parameters as client information.

Parameter	Required	Description
Client name	○	The name of the client.
URL of the client's logo		This logo is displayed on the consent screen and in other locations. If no URL is specified, the default logo is shown.
Redirect URIs	○	Register at least one redirect URI. You can register up to five redirect URIs. Redirect URIs must use https.
Scope	○	Register at least one scope. The scopes you need to call all Chatwork APIs are listed in the Appendix scope list.

2. Obtaining an authorization code

To obtain an authorization code, you need to access the consent screen via a web browser.

How to access the consent screen

The consent screen is displayed after you set the necessary query parameters and access the consent screen (URL: https://www.chatwork.com/packages/oauth2/login.php).
*If the login screen is displayed, enter your authentication information and log in. The web browser is then redirected to the consent screen.

Query parameters

Parameter	Specified Value	Required	Description
response_type	Strings	○	Currently, only “code” is supported.
client_id	Strings	○	Specify the client ID.
redirect_uri	Strings		Choose from the URLs set during client registration. For example: https://example.com/callback This is required if more than one redirect_uri is set, but optional if only one redirect_uri is set.
scope	Strings	○	rooms.all:read_write ● Appendix scope list
state	Strings		This is a character string that is used as a countermeasure against CSRF attacks. It is managed by associating it with the resource owner's session. When the resource owner approves or rejects authorization, the state string set here is set as the query parameter and the web browser is redirected to redirect_uri. After this, the state associated with the session is compared to the state string that was passed on as the query parameter. Always take countermeasures against CSRF attacks in production runs. For example: 343ab3341331218786ef
code_challenge	Strings		● code_challenge string format ○ String length should be 43 to 128 characters ○ String should correspond to [a-zA-Z0-9-._~]+ ● code_challenge value ○ code_challenge = URLSafe Base64(SHA256(code_verifier))) ○ code_verifier is used at token end point.
code_challenge_method	Strings		"S256" should be specified as a fixed value in order to use code_challenge.

For example: URL for displaying the consent screen

```
https://www.chatwork.com/packages/oauth2/login.php?
response_type=code
&redirect_uri=https://example.com/callback.php
&client_id=Lvo0YN92ga5kP
&state=811435b3683ae95c1cf3197deaf1bfe4b411f587
&scope=rooms.all:read_write%20users.profile.me:read
&code_challenge=jlkGAsNvHshJNC7uXSSmC2tALONajPdupVf3TScb7zk
&code_challenge_method=S256
```

Generating the authorization code

When the resource owner clicks the "Authorize" button on the consent screen, an authorization code is generated. The web browser is then redirected to redirect_uri after the authorization code is set as the query parameter.

For example: Redirect destinations when the authorization code is generated

- code
  - Authorization code with a validity period of 1 minute
- state
  - The value of the state specified when the consent screen is displayed

```
https://example.com/callback.php?
code=a2f0c1fe96af8c3a46fa0
&state=811435b3683ae95c1cf3197deaf1bfe4b411f587
```

Denying authorization

When the resource owner clicks the "Deny" button on the consent screen, authorization is denied, and the web browser is redirected to redirect_uri.

- error
  - Set to access_denied
- state
  - The value of the state specified when the consent screen is displayed

```
https://example.com/callback.php?
error=access_denied
&state=811435b3683ae95c1cf3197deaf1bfe4b411f587
```

Limitations of the login screen

Authentication using SAML is not supported via the dedicated OAuth2 login screen. Login using SAML authentication via the regular login screen, then go to the consent screen again.

- Login screen

3. Generating or regenerating an access token

Request

Endpoint

Use HTTP POST to access the following endpoint.

```
https://oauth.chatwork.com/token
```

Authentication

In accordance with Basic Authentication, specify the client ID as the user name and the client secret as the password, join these with a colon (:), and encode the character string with Base64. You can obtain this information via the client administration screen.

```
authorization: Basic THZvMFl1OO==
```

Format of the request body

```
application/x-www-form-urlencoded
```

Body parameters

When generating an access token with an authorization code:

Parameter	Specified Value	Required	Description
grant_type	Strings	○	Specify authorization_code.
code	Strings	○	Specify the authorization code (Authorization Code) for authorization.
redirect_uri	Strings		Specify the value that was specified when the consent screen is displayed. For example: https://example.com/callback This is required if more than one redirect_uri is set, but optional if only one redirect_uri is set.
code_verifier	Strings	required if code_challenge and code_challenge_method when authorization request	• String length should be 43 to 128 characters • String should correspond to [a-zA-Z0-9-._~]+

When regenerating an access token using a refresh token:

Parameter	Specified Value	Required	Description
grant_type	Strings	○	Specify refresh_token.
refresh_token	Strings	○	Specify the value of refresh_token that was generated when the token was generated.
scope	Strings		users.profile.me:read

Response

Format of the response body

```
application/json
```

Response body (identical whether a token is being generated or regenerated)

JSON Field Name	JSON Value	nullable	Ex	Description
access_token	String			The validity period is 15 minutes.
refresh_token	String	○		The validity period is 14 days. (However, this is subject to change)
token_type	String	○	Bearer	
expires_in	Int	○		The validity period of the token. (In seconds)
scope	String	○		
error	String	○	"invalid_grant"	Field that is returned when an error occurs.
error_description	String	○		Field that is returned when an error occurs.
error_uri	String	○		Field that is returned when an error occurs. Currently always null.

For example: Generating a token with the curl command

Request

```
curl  -v --user 'Lvo0YN92ga5kP:secret' \
-X POST -d 'grant_type=authorization_code \
&code=26d13798facc9a0ca05a8cb7246020f15a311 \
&redirect_uri=https://127.0.0.1/callback'  \
&code_verifier=5b0029bd34e559e0abe7a37051aa411398913fc3579e27bd963a2b9a647f12f58a335beeb4d
83a53a74ff1a6f99f6af385d2992c73beead39f57dcee95e0f954'  \
https://oauth.chatwork.com/token
```

Response

```
> POST /token HTTP/2
> Host: 127.0.0.1
> authorization: Basic THZvMFlOOTJnYTVrUDphYmNkZWZnaGlqa2xu
bW9wcXJzdHV2d3h5ejAxMjM0NTY3ODk=
> accept: */*
> content-length: 199
> content-type: application/x-www-form-urlencoded

< HTTP/2 200
< cache-control: no-store
< pragma: no-cache
< content-type: application/json
< content-length: 989

{
  "access_token": "eyJjdHkiOiJKV1QiLCJ0eXAiOiJKV1QiL
CJhbGciOiJSUzI1NiIsImtpZCI6ImlOUVh0dFR2RHZhcDVkSW
dWQiOiJodHRwczovL2FwaS5jaGF0d29yay5jb20iLCJzdWIi
BnRlQOXvXqymGijQXgqDOo3LLFY_k62OoPYAQ3UXkaum8
6Al-DJM6iC-043kBINbYLLPo0uwwsolmjRDG5zBzPC0GtcjXiLy4Gqg",
  "token_type": "Bearer",
  "expires_in": "1501138041000",
  "refresh_token": "86277ab4fd9d111bd3225215d96d6
22c9ae6810d82cd6d0e9530bf35adda67ab7d3c24e2a0
052e9d3b442ce212ca17ecf07ddbd8c3477aa3abde15e4ebcf7b53",
  "scope": "rooms.all:read_write"
}
```

4. Accessing a Chatwork API

Request

When sending a request to a Chatwork API, use the Bearer authentication scheme to set the access token (access_token) in the authorization request header field instead of using the usual Chatwork API authentication method.

```
authorization: Bearer eyJjdHkiOiJKV1QiLCJ0eXAiOiJKV1QiLCJh
bGciOiJSUzI1NiIsImtpZCI6ImlOUVh0dFR2RHZhcDVkSWpGQzA5Z
HZadHFXaGQ2WmFRb2pKenVuUS1vV28ifQ.eyJhdWQiOiJodHRwcz
ovL2FwaS5jaGF0d29yay5jb20iLCJzdWIiOiIzIiwiYWNjb3VudF9pZC
I6IjMiLCJzY29wZSI6WyJhbGwiXSwiaXNzIjoiaHR0cHM6Ly9vYXV0a
C5jaGF0d29yay5jb20iLCJleHAiOjE1MDExMzgwNDEsImlhdCI6MTU
wMTEzNzE0MSwianRpIjoiOTcwNDAwOWItNTdlNi00NDU5LTg5NzMt
Njc3ZmM5YjA5MjgyIiwiY2xpZW50X2lkIjoiTHZvMFlOOTJnYTVrUCJ9.
BIS8QvyTHz7KK_fnmvc0fa8NQDOWy7v8Ni0LvLyuROE5UEi7l_Hx
DT8tHLTQLELIm3jOw4SiW94KPYwduRL467vJ2j2eNT-zTkCXtEN
8pxbA0HtnBrtCcp0dRJEMnfBegzkoAe8BTB6gee3rrXy6sQcLb19
WBrrHNbjICFL0--SG3IvPanOzABqiNMqfScnasTtj7xtIaNpbxf8LDIH3E
F150Iif4BqSczJr-XppBTBYuP32UlBnRlQOXvXqymGijQXgqDOo3LLFY
_k62OoPYAQ3UXkaum86Al-DJM6iC-043kBINbYLLPo0uwwsolmjRD
G5zBzPC0GtcjXiLy4Gqg
```

Response

Please see the documentation for the endpoint of each Chatwork API.

Errors

When the access token has been revoked or is invalid, error information is set in the **www-authenticate** header field.

For example: When the access token has been revoked

```
HTTP/2 401
www-authenticate: Bearer error="invalid_token",
error_description="The access token expired"
```

For example: Accessing a Chatwork API using the curl command

```
curl -v -H 'authorization: Bearer eyJjdHkiOiJKV1QiLCJ0eXAiOiJKV1
QiLCJhbGciOiJSUzI1NiIsImtpZCI6ImlOUVh0dFR2RHZhcDVkSWpGQz
A5ZHZadHFXaGQ2WmFRb2pKenVuUS1vV28ifQ.eyJhdWQiOiJodHR
wczovL2FwaS5jaGF0d29yay5jb20iLCJzdWIiOiIxNzMiLCJhY2NvdW5
0X2lkIjoiMTczIiwic2NvcGUiOlsiYWxsIl0sImlzcyI6Imh0dHBzOi8vb2F1
dGguY2hhdHdvcmsuY29tIiwiZXhwIjoxNTAyMjUxNDU4LCJpYXQiOjE1
MDIyNDk2NTgsImp0aSI6IjRlNzg5ZTAzLTk2NjAtNDc4MC1hYThkLTV
mZjk2YmU0MzMyNSIsImNsaWVudF9pZCI6Ikx2bzBZTjkyZ2E1a1AifQ.
Y14Sr0SmtgwLegwWPMeQlPut2XmP74y3QdupCAN7Hc5Id10Qvgq-
csuYVxxAYStqZUO4sZ_j9SeE7-rqhuNowDMwqVaTGfDvAvtQLitPKD
Ub2g6x87c-lfffkJkIiL1xcH3lHrmQkBa_H81-_a3VFJila8hFptvygOp1
9OSDSrUIlcq6PeHlfNQtXjs2VFREQydQNE2cdLe68Nh5F5V4HX20C4
49MKNWK4ybwmFrnX-o9KgERaP1rjrCcWYrZ-lK8TquHti9XMfSaj71e
DkfPVOLyCOe1_zBEEH8NRFtN2OcVQWJFPy09rz1yw7a1YARdsU4
DukrhWvWcVxIad8ygA' \
'https://api.chatwork.com/v2/me'
```

```
> GET /v2/me HTTP/2
> Host: api.chatwork.com
> accept: */*
> authorization: Bearer eyJjdHkiOiJKV1QiLCJ0eXAiOiJKV1QiLCJ
hbGciOiJSUzI1NiIsImtpZCI6ImlOUVh0dFR2RHZhcDVkSWpGQzA5Z
HZadHFXaGQ2WmFRb2pKenVuUS1vV28ifQ.eyJhdWQiOiJodHRwc
zovL2FwaS5jaGF0d29yay5jb20iLCJzdWIiOiIxNzMiLCJhY2NvdW50
X2lkIjoiMTczIiwic2NvcGUiOlsiYWxsIl0sImlzcyI6Imh0dHBzOi8vb2F1
dGguY2hhdHdvcmsuY29tIiwiZXhwIjoxNTAyMjUxNDU4LCJpYXQiOjE
1MDIyNDk2NTgsImp0aSI6IjRlNzg5ZTAzLTk2NjAtNDc4MC1hYThkL
TVmZjk2YmU0MzMyNSIsImNsaWVudF9pZCI6Ikx2bzBZTjkyZ2E1a1
AifQ.Y14Sr0SmtgwLegwWPMeQlPut2XmP74y3QdupCAN7Hc5Id10
Qvgq-csuYVxxAYStqZUO4sZ_j9SeE7-rqhuNowDMwqVaTGfDvAvtQ
LitPKDUb2g6x87c-lfffkJkIiL1xcH3lHrmQkBa_H81-_a3VFJila8hFpt
vygOp19OSDSrUIlcq6PeHlfNQtXjs2VFREQydQNE2cdLe68Nh5F5V4
HX20C449MKNWK4ybwmFrnX-o9KgERaP1rjrCcWYrZ-lK8TquHti9X
MfSaj71eDkfPVOLyCOe1_zBEEH8NRFtN2OcVQWJFPy09rz1yw7a1
YARdsU4DukrhWvWcVxIad8ygA
>
< HTTP/2 200
< content-type: application/json; charset=utf-8
< content-length: 412
< connection: keep-alive
< x-ratelimit-limit: 300
< x-ratelimit-remaining: 299
< x-ratelimit-reset: 1502250163
< vary: Accept-Encoding,User-Agent
<
{"account_id":1, ...(略) }
```

Regarding the offline_access scope

Even when a resource owner is offline, the offline_access scope allows for persistent API access.

If this scope is not specified, it is assumed by the resource owner that the client will use the service while online, and after a certain period of time a refresh token will become unavailable if that is not the case, making it impossible to update the access token.
This is the default behavior and is the generally recommended usage, but it is not suitable for things such as bots where the resource owner may be absent. In such cases, it is possible to use the offline_access scope.

When the offline_access scope is specified, the refresh token remains valid indefinitely, unless it is revoked.
Until it is revoked, the access token can be renewed, even when the resource owner is absent.

Compared to online use, the refresh token can be used for long periods, so please give careful consideration to potential security risks before using.

Regarding PKCE

PKCE (Proof Key for Code Exchange by OAuth Public Clients) is an extended specification for OAuth2, which is considered to protect public clients from authorization code interception attack.There is a possibility where the authorization code is intercepted by attacker's application when the public client executes authorized code flow. PKCE prevent this type of attacks. Currently, confidential client is the only the client type you can register(you can also use PKCE for confidential client.). The public client is currently under development.

Appendix

A.List of Scopes

Name	API	Description
offline_access	All API	Allow permanent API access
users.all:read	users.profile.me:read users.status.me:read users.tasks.me:read	Access your account information
users.profile.me:read	GET /me	Access your profile information
users.status.me:read	GET /my/status	Access your unread messages
users.tasks.me:read	GET /my/tasks	Access your task lists
rooms.all:read_write	rooms.all:read rooms.all:write	Access and modify your chatroom's messages, tasks, files, notes and member information
rooms.all:read	rooms.info:read rooms.members:read rooms.messages:read rooms.tasks:read rooms.files:read	Access your chatroom's messages, tasks, files, notes and member information
rooms.all:write	rooms:write rooms.info:write rooms.members:write rooms.messages:write rooms.tasks:write	Modify your chatroom's messages, tasks, files, and member information

Name	API	Description
rooms:write	POST /rooms DELETE /rooms/{room_id}	Create new chatrooms and delete your chatrooms
rooms.info:read	GET /rooms GET /rooms/{room_id}	Access your chatrooms list
rooms.info:write	PUT /rooms/{room_id}	Modify your chatroom's information
rooms.members:read	GET /rooms/{room_id}/members	Access your chatroom's member information
rooms.members:write	PUT /rooms/{room_id}/members	Modify your chatroom's member information
rooms.messages:read	GET /rooms/{room_id}/messages GET /rooms/{room_id}/messages/{message_id}	Access your chatroom's messages
rooms.messages:write	POST /rooms/{room_id}/messages PUT /rooms/{room_id}/messages/{message_id} DELETE /rooms/{room_id}/messages/{message_id} PUT /rooms/{room_id}/messages/read PUT /rooms/{room_id}/messages/unread	Post messages to your chatrooms

Name	API	Description
rooms.tasks:read	GET /rooms/{room_id}/stasks GET /rooms/{room_id}/tasks/{task_id}	Access your chatroom's tasks
rooms.tasks:write	POST /rooms/{room_id}/tasks	Create tasks in your chatrooms
rooms.files:read	GET /rooms/{room_id}/files GET /rooms/{room_id}/files/{file_id}	Access file infomration in your chatrooms
contacts.all:read_write	contacts.all:read contacts.all:write	Access and modify your contact information
contacts.all:read	GET /contacts GET /incoming_requests	Access your contact information
contacts.all:write	PUT /incoming_requests/{request_id} DELETE /incoming_requests/{request_id}	Modify your incoming contact requests information

B. List of errors that can occur during authorization

Error Code	Error Description	Supplementary
1001	`response_type` parameter is missing.	The response_type parameter is not specified when the consent screen is displayed.
3001	The resource owner denied the request.	The resource owner refused permission on the login screen or consent screen.
4001	`token` response type is not supported.	An unsupported response_type is specified when the consent screen is displayed.
4002	`foo` response type is unknown.	An undefined response_type is specified when the consent screen is displayed.
5001	Scope is missing.	The scope is not specified.
5002	The scope is unknown.	An undefined scope is specified.
11000	`client_id` is missing.	The client_id is not specified.
13000	`redirect_uri` is missing.	A redirect_uri is not specified.
14000	The redirect URI is malformed.	The format of client_id is incorrect.
15000	The redirect URI is unregistered.	client_id is not registered.
18000	`code_challenge_method` is unsupported.	An unsupported code_challenge_method is specified.
19000	`code_challenge` is malformed.	The format of code_challenge is incorrect.
20000	`code_verifier` is malformed.	The format of code_verifier is incorrect.

Webhook

About Webhook

Chatwork Webhook sends you real-time notifications of events, such as sending and editing messages and you being mentioned, in chatrooms (group chat, direct chat, my chat) you are participating to a Webhook URL you specify.

By using Chatwork Webhook, you can receive real-time notifications without having to periodically call Chatwork API (by polling) to bring up events.

Please give it a try, as you will be able to easily create functions that can link with external services, such as an interactive bot that can operate inside of Chatwork.

Webhook URL can be set from the API control screen.

Request

Chatwork Webhook sends event notifications by giving HTTPS POST requests to a Webhook URL specified on Webhook control screen. POST requests have the following structure:

Common request header

Field Name	Value	Sample
content-type	application/json	content-type: application/json
user-agent	ChatWork-Webhook/<version>	user-agent: ChatWork-Webhook/1.0.0
x-chatworkwebhooksignature	Signature	x-chatworkwebhooksignature: 5202BF4...B02772F

Common request body

It is a JSON object with the following items:

Field Name	Type	Required	Description
webhook_setting_id	String	○	Set Webhool ID to notify this event.
webhook_event_type	String	○	Types of Webhook Event Object
webhook_event_time	Value	○	The time when the event was recorded on Chatwork Webhook system (epoch seconds).
webhook_event	JSON Object	○	Webhook event object suitable for webhook_event_type.

Sample request body

```
{
    "webhook_setting_id": "12345",
    "webhook_event_type": "mention_to_me",
    "webhook_event_time": 1498028130,
    "webhook_event":{
        "from_account_id": 123456,
        "to_account_id": 1484814,
        "room_id": 567890123,
        "message_id": "789012345",
        "body": "[To:1484814]What do you like to eat?",
        "send_time": 1498028125,
        "update_time": 0
    }
}
```

Response

- Always respond to an HTTPS request with a status code 200.
- The HTTPS POST Request sent from Chatwork Webhook will not be resent, even it fails.
- The maximum response body size is 2,048 bytes. Anything in excess will be considered as an error.

About the Failure Limit function for when the notification error rate is on the rise.

When an HTTPS POST Request causes an error for some reason, it is treated as a notification error. When the notification error rate becomes high, Chatwork Webhook automatically changes applicable Webhook settings to an “invalid” status. Users must explicitly validate his/her status. Please see Webhook editing screen for how to validate user status.

Signature verification for requests

In order to verify that the source of the request is indeed Chatwork, for each request, the signature must be verified on the user's server.

Verification process is as follows:

1. The digest value for the request body will be obtained through HMAC-SHA256 algorithm, using a byte string of a BASE64 decoded token as the secret key.

2. The string of BASE64 encoded digest value is verified that it matches the signature (x-chatworkwebhooksignature's header value). on the request header.

Token can be verified on Webhook editing screen.

Webhook Event Object

- Create message（webhook_event_type = "message_created"）
- Edit message（webhook_event_type = "message_updated"）
- Getting mentioned（webhook_event_type = "mention_to_me"）

Create message（webhook_event_type = "message_created"）

Field Name	Type	Required	Description
message_id	String	○	Message ID
room_id	Value	○	Chatroom ID, where the message was sent.
account_id	Value	○	Account ID, from which the message was sent.
body	String	○	The content of the message.
send_time	Value	○	Time when the message was sent (epoch seconds).
update_time	Value	○	The last time the message was edited (epoch seconds). *Warning: the value is 0, when the message was created.*

Sample event object when creating a message:

```
{
    "message_id": "789012345",
    "room_id": 567890123,
    "account_id": 1484814,
    "body": "Please prepare your presentation slides up to 3 pages",
    "send_time": 1498028120,
    "update_time": 0
}
```

Edit message（webhook_event_type = "message_updated"）

Message edit event has the same structure as Create message event.

Getting mentioned（webhook_event_type = "mention_to_me"）

Field Name	Type	Required	Description
from_account_id	Value	○	Account ID of the user that mentioned another user.
to_account_id	Value	○	Account ID of the user that got mentioned.
room_id	Value	○	Chatroom ID, in which the user was mentioned.
message_id	String	○	Message ID, in which the user was mentioned.
body	String	○	The content of the message in which the user was mentioned.
send_time	Value	○	Time when the message was sent (epoch seconds).
update_time	Value	○	The last time the message was edited (epoch seconds). *Warning: the value is 0, when the message was created.*

Sample event when getting mentioned:

```
{
    "from_account_id": 1234567890,
    "to_account_id": 1484814,
    "room_id": 567890123,
    "message_id": "789012345",
    "body": "[To:1484814] Can you prepare the presentation slide?",
    "send_time": 1498028125,
    "update_time": 0
}
```

URLs you can use as Webhook URL

There are some restrictions for the URLs you can use for Webhook URL

- It should be starting with "https://".
- If the domain is IDN, it should be converted with ToASCII (4.1) based on RFC3490.
- It should be encoded based on RFC3986.