TeamViewer API

Documentation

Version 1.0.1

Table of Contents

1 Introduction

1.1 Definitions

Name	Description
`Supporter`	A user who provides support for the end customer. This user must have a TeamViewer account.
`End Customer`	A user who does not have to have a known TeamViewer account. The user is in most cases a customer of the company who is using the API.
`Client`	The application (or user, if the HTTP requests are typed in manually) that uses the API.

1.2 REST

The TeamViewer API is a REST API which uses the already existing HTTP methods to create (`POST`), read (`GET`), change (`PUT`) or delete (`DELETE`) single items or a collection of items. The following table shows the general use cases for these HTTP methods.

	GET	POST	PUT	DELETE
`Collection`	retrieve list of items in this collection	create new item in this collection	-	-
`Single item`	retrieve item data	-	changes the item	deletes this item

The basic URI scheme for all API functions is: `https://host/path/to/resources[/id][/verb][?param1=value1]`

Parameters in the URI are only allowed for GET and DELETE. Generally there should be no need for any parameters for DELETE, though. POST and PUT need to have the parameters in the body formatted as JSON or XML.

1.3 IDs

IDs are prefixed with a type in order to make them more distinguishable. The following types are used:

- "u" - user ID
- "g" – group ID
- "s" – session code

Reports use a GUID and are not prefixed.

1.4 Date format

All dates and times follow the ISO 8601. They should have the following format: `YYYY-MM-DD"T"HH:MM:SS"Z".` Times are always in UTC unless stated otherwise.

Example	`2013-02-21T13:42:55Z = 21st February 2013, 13:42:55 UTC`

1.5 Number format

Decimal numbers are returned in US English format, using a point as decimal separator. Digits are never grouped by a delimiter.

Example	`12345.67`

2 User permissions

2.1 Permission names in API and Management Console

The following table shows the relation between user permissions as they can be set in the Management Console and the technical name that is used for the API.

Name in API	Name in Management Console
ManageAdmins	Manage administrators and company settings
ManageUsers	Manage users
ShareOwnGroups	Allow group sharing
ViewAllConnections	View all connections
ViewOwnConnections	View own connections
EditConnections	Edit logged connections
DeleteConnections	Delete logged connections
EditFullProfile	Allow full profile modification
ManagePolicies	Manage & Assign policies
AssignPolicies	Assign policies
AcknowledgeAllAlerts	View & acknowledge all alerts
AcknowledgeOwnAlerts	View & acknowledge own alerts
ViewAllAssets	View all assets

Name in API	Name in Management Console
ViewOwnAssets	View assets
EditAllCustomModuleConfigs	Manage all customizations
EditOwnCustomModuleConfigs	Manage own customizations
None	<no permission is set in console>

2.2 Permission dependencies

Some permissions are depending on each other and cannot be set individually, e. g. if a user has the permission to ManageAdmins he also must have the permission to ManageUsers. When these permissions are set in the Management Console, the other permissions are set automatically. When setting permissions through the API using a PUT or POST command, all permissions must be included explicitly. The following table shows the dependencies.

Name in API	To set this right through the API, these rights also have to be set
None	-
ManageAdmins	ManageUsers, ShareOwnGroups, EditFullProfile, ViewAllConnections, ViewOwnConnections, EditConnections, DeleteConnections, ManagePolicies, AssignPolicies, AcknowledgeAllAlerts,AcknowledgeOwnAlerts, ViewAllAssets, ViewOwnAssets, EditAllCustomModuleConfigs, EditOwnCustomModuleConfigs
ManageUsers	ShareOwnGroups, EditFullProfile, ViewAllConnections, ViewOwnConnections, EditConnections, DeleteConnections, ManagePolicies, AssignPolicies, AcknowledgeAllAlerts, AcknowledgeOwnAlerts, ViewAllAssets, ViewOwnAssets, EditAllCustomModuleConfigs, EditOwnCustomModuleConfigs
ShareOwnGroups	-
ViewAllConnections	ViewOwnConnections
ViewOwnConnections	-
EditConnections	-
DeleteConnections	-

Name in API	To set this right through the API, these rights also have to be set
EditFullProfile	-
ManagePolicies	AssignPolicies, AcknowledgeAllAlerts, AcknowledgeOwnAlerts
AssignPolicies	AcknowledgeAllAlerts, AcknowledgeOwnAlerts
AcknowledgeAllAlerts	AcknowledgeOwnAlerts
AcknowledgeOwnAlerts	-
ViewAllAssets	ViewOwnAssets
ViewOwnAssets	-
EditAllCustomModuleConfigs	EditOwnCustomModuleConfigs
EditOwnCustomModuleConfigs	-

3 API Functions

3.1 Authentication (OAuth 2.0)

For more information about OAuth 2.0, see http://oauth.net/2/ and the official specification at http://tools.ietf.org/html/rfc6749

Names used by the RFC and their meaning for the TeamViewer API:

- **resource owner** – The user behind a TeamViewer account who wants to access their resources through the API.
- **resource server** – Our servers where the API runs.
- **client** – The application, plug-in, script or user who is making the API HTTP requests.
- **authorization server** – In our case that's the same servers that run the rest of the API.
- **client ID** – A unique ID to identify the application that wants to use the TeamViewer API.
- **client secret –** A unique string only known to the creator of the client ID.
- **authorization code** – Code used during the OAuth process to prove that an authorization request was granted in the Management Console.
- **access token** – A token that has to be used to access any API function (except those explicitly marked as not requiring any access tokens).
- **refresh token** – A token that can be used once to obtain a new access token and a new refresh token.

3.1.1 Registering an application in the Management Console

Before using any API functionality you need to register an application in the Management Console. When you register the application you have to specify if you want to use it for your own account only (private application, also referred to as “Script”) or if you want to create an application to be used by any TeamViewer user (public application, also referred to as “App”). In both cases you also specify if the application will have access to the data of one single account or to the data of the entire company.

Client/Application-Types:

	Script	App
User Access	Access token is created that can only be used to access the user who created the application.	Client ID is created when creating the application. The Client ID can be used with OAuth to create an access token for the user granting access.

	Script	App
Company Access	Access token is created that can be used to access the company of the user (=company admin) who created the application.	Client ID is created when creating the application. The Client ID can be used with OAuth to create an Access-Token to access the company of the user (=company admin) granting access.

When you register an application for your own use only, you will get an access token that can be used directly for any API function that requires it. When you register the application for others to use as well, you will get a Client ID. This Client ID is used in the OAuth process described below. At the end of this process the application will also have an access token that must be used by the other API functions. This access token is tied to the account/company that **uses** the application, not the company that **created** the application.

If you are using OAuth in your application and the application was registered for company use, the user who grants access to your application needs to be an administrator. The user who registered the application does not have to be in a company however.

3.1.2 Access Token Permissions and Scopes

Access tokens can be either issued for a single user or for a whole company. Company access tokens have to be created by an administrator of that company. Besides this distinction access tokens also have a number of permissions attached to them. This is called the scope of the access token.

The following table shows all available scopes.

API-Function	Scopes
Account (user level access only)	Account.Create, Account.Read, Account.ReadEmail, Account.Modify, Account.ModifyEmail, Account.ModifyPassword, Account.ReadOnlineState
Groups	Groups.Create, Groups.Read, Groups.Modify, Groups.Share, Groups.Delete
Users (company level access only)	Users.CreateUsers, Users.CreateAdministrators, Users.Read, Users.ModifyUsers, Users.ModifyAdministrators
Sessions	Sessions.Create, Sessions.ReadAll, Sessions.ReadOwn, Sessions.ModifyAll, Sessions.ModifyOwn
Connections	Connections.Read, Connections.Modify, Connections.Delete

3.1.3 The authorization process

When using private Script Tokens, there is no need for an authorization process. Access to the account/company data through the TeamViewer API is defined when creating the token. The data that is accessed is the account or company data of the user creating the token. Note that a Script Token is still valid after changing the user's password.

For public Apps the case is different. Because these applications can be used by other TeamViewer users, access to their data is controlled via OAuth 2.0. We distinguish between application with access to user level data and company level data.

3.1.3.1 User level access

If a user starts an app that requires user level access for the first time, TeamViewer will ask the user to grant a set of permissions to the app. This set of permissions was specified when creating the application. The permissions are checked against the rights of the current user. If the application asks for permissions that exceed the rights of the user (e. g. the application wants to edit connection report entries whereas the user is only allowed to view them), the permissions in question are highlighted and a warning is displayed that some parts of the application may not behave as intended because of the lacking user rights.

In any case, the user may either choose to deny or to grant access to the application. If access is granted, the app can access the user's data, as long as the user's permissions allow. If user rights are changed later, the application may be able to access more data.

3.1.3.2 Company level access

To grant company level access to an application the user needs to have all required permissions. Unlike user level access it is not possible to grant access when some permissions are missing.

Also unlike user level access the rights of the user are not relevant for the app any more after granting access.

3.1.4 GET /api/v1/oauth2/authorize (authorization endpoint)

Parameters

- **`response_type`** – Must be `code`.
- **`client_id`** – Client ID, a unique string that identifies the application.
- **`redirect_uri`** (*optional*) – URI to redirect to once access has been granted. If this parameter is left out the code that would have been returned here is shown in the browser and has to be copied manually to the application. If a fixed redirect URI is specified for the provided client ID, the value of this parameter must match it.
- **`state`** (*optional*) – Can be set if needed, and will be returned to the callback URI if it was set here.

Return values

Not applicable here, because this needs to be opened in a browser and will return a website. The other values will be added to the `redirect_uri` once access has been granted.

- **`code`** – Code that can be used to get an access token.
- **`state`** – same as the one provided as parameter.

Description

Requests an authorization code from the server. This code is only valid for 10 minutes and should be used to get an access token. This is the only function that should not be called directly from a 3rd party application but it should be opened in a browser and return a website where the user can grant access to the 3rd party application.

Example

Request:

```
GET /api/v1/oauth2/authorize?response_type=code&client_id=12333-
133Ea4Hdf3e9ec0543fX&redirect_uri=https%3A%2F%2Fclient%2Eexample%2Ecom%2Fcb HTTP/1.1
```

3.1.5 POST /api/v1/oauth2/token (token endpoint)

Parameters

Parameters must be inside the body of the request and encoded with the "application/x-www-form-urlencoded" format. This is the only exception where the body is not JSON or XML. There are two different requests for this URI, one to retrieve an access token using an authorization code and one using a refresh token.

Authorization-Code Grant:

- **`grant_type`** – Must be `authorization_code`.
- **`code`** – Authorization code acquired from the /oauth2/authorize page.
- **`redirect_uri`** (*optional*) – Must be the same value as in the previous call to /oauth2/authorize. If no **`redirect_uri`** was given none should be added here as well.
- **`client_id`** – Client ID, a unique string that identifies the application.
- **`client_secret`** – The client secret, which is known only to the creator of the application.

Refresh-Token:

- **`grant_type`** – Must be `refresh_token`.
- **`refresh_token`** – Refresh-token from a previous call.
- **`client_id`** – Client ID, a unique string that identifies the application.
- **`client_secret`** - The client secret, a unique string known only to the creator of the application.

Return values

- **`access_token`** – Access token to use with all further API calls.
- **`token_type`** – Authentication-method used for this access token, currently only `bearer` is used.
- **`expires_in`** – Time in seconds until the access token expires and needs to be refreshed.
- **`refresh_token`** – Refresh-Token that needs to be used to get a new access token when the old access token expires. Requesting a new access token will also create a new refresh token.

Description

Requests a new access token, either by using the code from a previous authorization step or by using an existing refresh token. Access tokens have a limited lifetime of 1 day. The response always has the Cache-Control and Pragma fields (see example below). Refresh tokens can only be used once. For this request the Content-Type header should be set to `application/x-www-form-urlencoded` (as per OAuth 2 specification), however JSON/XML also works.

Example

Request (Authorization code grant):

```
POST /api/v1/oauth2/token HTTP/1.1
    Host: webapi.teamviewer.com
    Content-Type: application/x-www-form-urlencoded

grant_type=authorization_code&code=SplxlOBeZQQYbYS6WxSb&redirect_uri=https%3A%2F%2Fc
lient%2Eexample%2Ecom%2Fcb&client_id=12333-133Ea4Hdf3e9ec0543fX
```

Response (Authorization code grant):

```
HTTP/1.1 200 OK
    Content-Type: application/json
    Cache-Control: no-store
    Pragma: no-cache

 {"access_token":"54213-2YotnFZFEjr1zCsicMWp",
  "token_type":"bearer",
  "expires_in":3600,
  "refresh_token":"12854-zv3JOkF0XG5Qx2TlKWIA"}
```

Requests using the access token

All API requests need to include the "Authorization" header if the API function requires an access token.

Example

```
GET /api/v1/users?online=1 HTTP/1.1
  Host: webapi.teamviewer.com
  Authorization: Bearer 54213-2YotnFZFEjr1zCsicMWp
```

All examples in the following sections will have this header omitted but if an access token is required the **`Authorization`** header field needs to be added to the request.

If no access token is given in the header, or the access token is past its expiration date, the return will have a WWW-Authenticate header field.

Response for no access token, but access token required:

```
HTTP/1.1 401 Unauthorized
  WWW-Authenticate: Bearer
```

Response for expired access token:

```
HTTP/1.1 401 Unauthorized
  WWW-Authenticate: Bearer

{ "error" : "token_expired",
  "error_code" : 1,
  "error_description" : "The access token expired" }
```

3.2 Ping

3.2.1 GET /api/v1/ping

Parameters

None

Return values

- **token_valid** – Is set to `true` if the provided access token is OK and the message is signed correctly. In all other cases the value is set to `false`.

Authentication

Access tokens are optional but will be verified if provided. Scope: None required.

Description

This function can be used to check if the API is available. It can also be used to verify if the token is valid.

Example

Request

```
GET /api/v1/ping
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{"token_valid":false}
```

3.3 Account Management

3.3.1 GET /api/v1/account (retrieve account data)

Parameters

None.

Return values

- **name** – The name of the user.
- **email** – The associated email address. Only returned if access token has the “Account.ReadEmail” scope.
- **userid** – The user ID.
- **company_name** (optional) – The name of the company that the user belongs to.

Authentication

User access token. Scope: Account.Read and (optionally) Account.ReadEmail.

Description

Retrieves account information of the account associated with the access token.

Example

Request

```
GET /api/v1/account
```

Response

```
HTTP/1.1 200 OK

Content-Type: application/json

{
   "userid": "u1234567",
   "email": "jdoe@example.com",
   "name": "John Doe",
   "company_name": "John's Company"
}
```

3.3.2 PUT /api/v1/account (change account data)

Parameters

- **`name`** (*optional*) – The name of the user.
- **`email`** (*optional*) **–** The associated email address. Note that changing the email address needs to be confirmed in the Management Console first before changes take effect.
- **`password`** (*optional*) **-** A new password for this account. If password is set, **`oldpassword`** must be set to the correct value of the old password.
- **`oldpassword`** (*optional*) - The old password of this account

Return values

None.

Authentication

User access token. Scope:Account.Modify, Account.ModifyEmail or Account.ModifyPassword..

Description

Changes account information of the account associated with the access token.

Example

Request

```
PUT /api/v1/account
  Content-Type: application/json

{ "name" : "John Locke" }
```

Response

```
HTTP/1.1 204 No Content
```

3.4 User Management

3.4.1 GET /api/v1/users (retrieve list of users)

Parameters

- **email** (*optional*) – Lists the user that has an exact match of the given email address.
- **name** (*optional*) – List all users that have the given string as part of their user name.
- **permissions** (*optional*) – List all users with certain permissions. Multiple permissions can be separated by comma and only users having all the permissions will be listed in that case. (See 2.1 for a list of possible values).
- **full_list** (*optional*) – `true`: list contains all info fields about the users, `false` (*default*): list contains only minimal info about the users

Return values

For the minimal list:

- **id** – user ID, needed to access/modify that user
- **name** – name of the user

For the full list (additionally):

- **permissions** (*optional*) – Comma-separated list of permissions that this user has. (See 2.1 for a list of possible values).
- **active** – true if the account is active, false otherwise.

Authentication

Company access token. Scope: Users.Read

Description

Lists all users in a company. The list can be filtered with additional parameters. The function can also return a list containing all information about the users. This data is the same as when using GET /users/uID for each of these users.

Example

Request

```
GET /api/v1/users?online=true&full_list=true
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json
```

```
{ "users" : [
 { "id" : "u1234567",
   "name" : "Mighty Administrator",
   "permissions" : "ManageAdmins, ManageUsers, ShareOwnGroups, EditFullProfile,
ViewAllConnections, ViewOwnConnections, EditConnections, DeleteConnections,
ManagePolicies, AssignPolicies, AcknowledgeAllAlerts,AcknowledgeOwnAlerts,
ViewAllAssets, ViewOwnAssets, EditAllCustomModuleConfigs,
EditOwnCustomModuleConfigs",
   "online": true
 },
 { "id" : "u2345678",
   "name" : "John Doe",
   "permissions" : "EditFullProfile",
   "online" : true
 }]
}
```

3.4.2 POST /api/v1/users (create new company member)

Parameters

- **email** – Email of that user. Will be used for login.
- **password** (*optional*) – Password for the user. Will be used for login. If omitted, users will need to set their own password before logging in.
- **permissions** (*optional*) – Comma-separated list of permissions that this user has. See 2.2 for valid values and combinations. If omitted the following default permissions will be set: `ShareOwnGroups`, `ViewOwnConnections`, `EditConnections`, `EditFullProfile`
- **name** – Name of the new user.
- **language** – Language code for the user. Will be used for the welcome email.

Return values

HTTP status code 201 on success and a JSON containing the same data as GET /users/uID. The new URI for that user will also be returned as part of the HTTP header.

Authentication

Company access token. Scope: Users.CreateUsers or Users.CreateAdministrators.

Description

Creates a new user for the company. The data for the new user will be returned as response to the POST. This should be the same as **GET /users/uID**, except that it will include the id as well. You will need to have the scope Users.CreateAdministrators to set the permissions `ManageUsers` or `ManageAdmins`.

Example

Request

```
POST /api/v1/users
  Content-Type: application/json

{ "email" : "foo@example.com",
  "password" : "abc!de#f3g2h3",
  "name" : "Ted",
  "language" : "en",
  "permissions" : "EditFullProfile" }
```

Response

```
HTTP/1.1 201 Created
  Content-Type: application/json
  Location: https://webapi.teamviewer.com/api/v1/users/u456789

{
  "id" : "u456789",
  "name" : "Ted",
  "permissions" : " EditFullProfile ",
  "online" : false
}
```

3.4.3 GET /api/v1/users/<uID> (get information for one user)

Parameters

None

Return values

This is the same as the full list for GET /users .

- **`id`** – User ID. Should be the same as in the URL.
- **`name`** – name of the user.
- **`permissions`** (*optional*) – String list of permissions that this user has. See 2.1 for valid values.
- **`active`** – `true` if the account is active, `false` otherwise.

Authentication

Company access token. Scope: Users.Read.

Description

Returns the information for a single user. The information is the same as when using **GET /users?full_list=1**.

Example

Request

```
GET /api/v1/users/u123
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{
  "id" : " u123",
  "email" : "user@example.com",
  "name" : "John Doe",
  "permissions" : "EditFullProfile",
  "online" : true
}
```

3.4.4 PUT /api/v1/users/<uID> (modify user information)

Parameters

- **email** (*optional*) – New Email of the user. A verification Email will be sent to the new address.
- **name** (*optional*) – Real name of the user.
- **permissions** (*optional*) – Comma-separated list of permissions that this user has. See 2.2 for valid values and combinations.
- **password** (*optional*) – Assign a new password for this user.
- **active** (*optional*) – Activates or deactivates an account.

Return values

HTTP status code 204 (No Content) on success.

Authentication

Company access token. Scope: Users.ModifyUsers or Users.ModifyAdministrators.

Description

Changes information for a selected user. Only the parts that need to be changed are needed in the request body.

Security-Warning: An attacker can gain access to a user account either by changing the email (+password reset) or by changing the password if he can steal the company access token. This makes the company access token equivalent to email and password for ALL company accounts with which an attacker can get the full Computer & Contacts list (and not just what is available over the API).

Example

Request

```
PUT /api/v1/users/u123
  Content-Type: application/json

{ "name" : "John Locke" }
```

Response

```
HTTP/1.1 204 No Content
```

3.5 Group Management

3.5.1 Accessing groups from different users in a company

Important note: If you are using a company access token, you can use all the functions below but have to prefix them with a user-location. So "**GET /groups**" for example becomes "**GET /users/<uID>/groups**".

3.5.2 GET /api/v1/groups (list all available groups)

(GET /api/v1/users/<uID>/groups for company access token)

Parameters

- **name** (*optional*) – Group name or part of the group name
- **shared** (*optional*) – `True`: list only shared groups, i. e. groups where the current user is not the owner. `False`: list only not shared groups, i. e. groups owned by the current user. If left out both types will be in the list.

Return values

- **groups** – List of groups.
  - **id** – Group ID
  - **name** – Name of the group.
  - **shared_with** (*optional*) – List of users who this group is shared with. Only available if the user is owner of the group. Can be omitted if empty.
    - **userid** – User ID of the user the group is shared with.
    - **name** – Name of the user the group is shared with.
    - **permissions** – Access-permissions of the user on this group. Either `read` or `read-write`.
    - **pending** – `true` if the user hasn't accepted the shared group yet, otherwise the field can be omitted.
  - **owner** (*optional*) – Owner of this group. Omitted if the owner is the current user.
    - **userid** – User ID of the owner of this group.
    - **name** – Name of the owner of this group.
  - **permissions** – `read`, `readwrite` or `owned`.

Authentication

User or company access token. Scope: Groups.Read.

Description

Returns a list of groups.

Example

Request

```
GET /api/v1/groups?name=Test
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{ "groups" : [
  { "id" : "g53235",
    "name" : "Testing",
    "shared_with" : [
    { "userid" : "u631645",
      "name" : "Ted",
      "permissions" : "read"}],
    "permissions" : "owned"
  },
  { "id" : "g425123356",
    "name" : "Test",
    "owner" : {
      "userid" : "u814464403",
      "name" : "Tester" },
    "permissions" : "readwrite"
  }]
}
```

3.5.3 POST /api/v1/groups (create a new group)

(POST /api/v1/users/<uID>/groups for company access token)

Parameters

- **name** – Name of the new group.

Return values

- **id** – Group ID of the newly created group.
- **name** – Name of the new group. This should be the same parameter as the input parameter.
- **permissions** – Will always be owned, because the group is not yet shared at this point.

Authentication

User or company access token. Scope: Groups.Create.

Description

Creates a new group and returns its info.

Example

Request

```
POST /api/v1/groups
  Content-Type: application/json

{ "name" : "Test" }
```

Response

```
HTTP/1.1 201 Created
  Content-Type: application/json
  Location: https://webapi.teamviewer.com/groups/g425123356

{
  "id" : "g425123356",
  "name" : "Test",
  "permissions" : "owned"
}
```

3.5.4 GET /api/v1/groups/<gID> (get group details)

(GET /api/v1/users/<uID>/groups/<gID> for company access token)

Parameters

None.

Return values

- **id** – Group ID
- **name** – Name of the group.
- **shared_with** (*optional*) – List of users who this group is shared with. Only available if the user is owner of the group. Will be omitted if empty.
  - **userid** – User ID of the user the group is shared with.
  - **permissions** – Access-permissions of the user on this group. Either `read` or `readwrite`.
- **owner** (*optional*) – Owner of this group. Will be omitted if the owner is the current user.
  - **userid** – User ID of the owner of this group.
  - **name** – Name of the owner of this group.
- **permissions** – Access permissions for the current user. `read`, `readwrite` or `owned`.

Authentication

User or company access token. Scope: Groups.Read.

Description

Returns info for one group.

Example

Request

```
GET /api/v1/groups/g425123356
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{
  "id" : "g425123356",
  "name" : "Test",
  "owner" : {
      "userid" : "u814464403",
      "name" : "Tester" },
  "permissions" : "owned"
}
```

3.5.5 PUT /api/v1/groups/<gID> (change group details)

(PUT /api/v1/users/<uID>/groups/<gID> for company access token)

Parameters

- **name** – New name of the group.

Return values

HTTP status code 204 if change succeeded.

Authentication

User or company access token. Scope: Groups.Modify.

Description

Changes a group. Right now only the name can be changed.

Example

Request

```
PUT /api/v1/groups/g425123356
  Content-Type: application/json

{ "name" : "Test 123" }
```

Response

```
HTTP/1.1 204 No Content
```

3.5.6 DELETE /api/v1/groups/<gID> (delete a group)

(DELETE /api/v1/users/<uID>/groups/<gID> for company access token)

Parameters

None

Return values

HTTP status code 200 on success.

Authentication

User or company access token. Scope: Groups.Delete

Description

Deletes an existing group. If the group is not owned, but only shared with the user's account it will just be unshared.

Example

Request

```
DELETE /api/v1/groups/g425123356
```

Response

```
HTTP/1.1 200 OK
```

3.5.7 POST /api/v1/groups/<gID>/share_group (share group with other user(s))

(POST /api/v1/users/<uID>/groups/<gID>/share_group for company access token)

Parameters

- **users** – List of users with whom the group will be shared.
  - **`userid`** – User ID of one of the users you want to share the group with.
  - **`permissions`** – Access-permissions of the user on this group. Either `read` or `readwrite`.

Return values

HTTP status code 200.

Authentication

User or company access token. Scope: Groups.Share.

Description

Shares a group with the given users. Will not change the share state with other users, but it is possible to overwrite the permissions for existing shares.

Example

Request

```
POST /api/v1/groups/g425123356/share_group
 Content-Type: application/json

{ "users" : [
  { "userid" : "u33516235",
    "permissions" : "read" },
  { "userid" : "u51235",
    "permissions" : "readwrite" }]
}
```

Response

```
HTTP/1.1 200 OK
```

3.5.8 POST /api/v1/groups/<gID>/unshare_group (unshare a group from certain users)

(POST /api/v1/users/<uID>/groups/<gID>/unshare_group for company access token)

Parameters

- **users** – List of User IDs that this group should not longer be shared with.

Return values

HTTP status code 200

Authentication

User or company access token. Scope: Groups.Share.

Description

Unshares a group from certain users.

Example

Request

```
POST /api/v1/groups/g425123356/unshare_group
  Content-Type: application/json

{ "users" : [ "u33516235", "u51235" ]}
```

Response

```
HTTP/1.1 200 OK
```

3.6 Session Management

3.6.1 GET /api/v1/sessions (list session codes)

Parameters

- **groupid** (*optional*) – Filter by group id.
- **assigned_userid** (*optional*) – Filter by assigned_userid.
- **state** (*optional*) – State of the session. Can be `open` or `closed`. By default only open sessions will be selected. States can be combined with a comma.
- **full_list** (*optional*) – `true`: Return all information for the sessions. This is `false` by default.
- **offset** (*optional*) – Can contain a session code from a previous request. The returned list will contain session codes after the one specified as offset.

Return values

For the minimalistic list (**full_list**=`false`):

- **sessions** – List of session codes.
  - **code** – Session code.
  - **state** – State of the session. Can be "open" or "closed".
  - **groupid** – Group ID where this session is stored under.
- **sessions_remaining** – Number of session codes left after the ones returned here. Will be omitted if there are no further session codes.
- **next_offset** – Offset that can be used to get the next 1000 session codes.

For the full list (**full_list=true**):

- **waiting_message** – Message displayed to the waiting end customer.
- **description** – Description for this session code.
- **end_customer** – End Customer info
  - **name** – Name of the end customer.
  - **email** – Email of the end customer.
- **assigned_userid** – User ID of the user this session is assigned to. If session is unassigned, value is “u0”.
- **assigned_at** – Date when the last user was assigned.
- **end_customer_link** – Link for the end customer.
- **supporter_link** – Link for the supporter.
- **custom_api** – Custom fields, this stores any arbitrary string but is only available to API functions. It is limited to 4000 characters.
- **created_at** – Date when the session code was created.
- **valid_until** – Date up to which the session code is valid.
- **closed_at** – Date when the session was closed.

Authentication

User or company access token. Scope: Sessions.ReadAll or Sessions.ReadOwn.

Description

Lists sessions. If no filters are given it will list all sessions in the active account (user access token) or all sessions from all accounts (company access token). A single request will return a maximum of 1000 session codes. To get the next 1000 session codes, repeat the same request with the offset parameter set to the value from **next_offset**. Session codes will be sorted by the **created_at** date from new to old.

Example

Request

```
GET /api/v1/sessions?groupid=g425123356&full_list=true
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{"sessions" : [
  {"code" : "s31-328-542",
   "groupid" : "g425123356",
   "description" : "Hello, I have an issue with my printer, can you please assist?",
   "end_customer" : { "name" : "Peter Niedhelp", "email" : "helpme@example.com" },
   "assigned_userid" : "u7254190",
   "end_customer_link" : "https://get.teamviewer.com/...",
   "supporter_link" : "https://get.teamviewer.com/...",
   "custom_api" : "{ "ticket_id" : "535824" }"
  }, …]
}
```

3.6.2 POST /api/v1/sessions (create new session code)

Parameters

- **valid_until** (*optional*) – Date up to which the session code is valid. Will default to 24h from now when no date is given.
- **groupid** (*partially required*) – ID of the group the session will be inserted into. The group can be in a different account when the API user has access to it. Either this or **groupname** must be set. For applications with company level access, this parameter is required.
- **groupname** (*partially required*) – Name of the group that the session code will be inserted into. Either this or the **groupid** parameter must be set. If no group exists with that name a new group will be created. If both **groupid** and **groupname** are set, both have to point to the same group. Otherwise an error is returned.
- **waiting_message** (*optional*) – Message displayed to the waiting end customer.
- **description** (*optional*) – Description for the new session code.
- **end_customer** (*optional*) – End customer info.
  - **name** (*optional*) – Name of the end customer. Maximum length is 100 characters.
  - **email** (*optional*) – Email of the end customer. Maximum length is 254 characters.
- **assigned_userid** (*optional*) – User ID of the user this session code will be assigned to. If not set or set to `u0`, session is unassigned.
- **custom_api** (*optional*) – Custom fields, this stores any arbitrary string but is only available to API functions. It is limited to 4000 characters.

Return values

- **code** – Newly created session code.
- **state** – State of the session. Can be `open` or `closed`.
- **groupid** – Group ID where this session is stored in.
- **waiting_message** – Message displayed to the waiting end customer.
- **description** – Description for this session code.
- **end_customer** – End customer info
  - **name** – Name of the end customer.
  - **email** – Email of the end customer.
- **assigned_userid** – User ID of the user this session code is assigned to. If the session code is not assigned the value will be `u0`.

- **assigned_at** – Date when the last user last assigned.
- **end_customer_link** – Link for the end customer.
- **supporter_link** – Link for the supporter.
- **custom_api** – Custom fields, this stores any valid JSON/XML object (max 4000 characters) and is only visible for API users.
- **created_at** – Date when the session code was created.
- **valid_until** – Date when up to which the session code is valid.

Authentication

User or company access token. Scope: Sessions.Create.

Description

Creates a new session code. A session code will always be stored in a group, so either the **groupid** or **groupname** parameter must be set. Session codes will expire after 24h if no **valid_until** date is set.

Example

Request (with user/company access token)

```
POST /api/v1/sessions
   Content-Type: application/json
   Authorization: Bearer 54321-abcdefghijklmnopqrstuvwxyz

{"groupid" : "g425123356",
 "description" : " I have aproblemwith myspace bar.",
 "end_customer" : { "name" : "Max" }
}
```

Response

```
HTTP/1.1 201 Created
  Content-Type: application/json
  Location: https://webapi.teamviewer.com/api/v1/sessions/s12-345-678

{"code" : "s12-345-678",
 "state" : "open",
 "groupid" : "g425123356",
 "end_customer" : { "name" : "Max" },
 "description" : "I have aproblemwith myspace bar.",
 "assigned_userid" : "u7254190",
 "end_customer_link" : "http://get.teamviewer.com/s12345678",
 "supporter_link" : "http://get.teamviewer.com/s12345678-asfg1234asfg",
 "valid_until" : "2013-10-30T12:03:29Z"
}
```

3.6.3 GET /api/v1/sessions/<code> (get info about a certain session code)

Parameters

None

Return values

- **code** – Session code.
- **state** – State of the session. Can be `open` or `closed`.

- **groupid** – Group ID where this session is stored in.
- **waiting_message** – Message displayed to the waiting end customer.
- **description** – Description for this session code.
- **end_customer** – End customer info
  - **name** – Name of the end customer. Maximum length is 100 characters.
  - **email** – Email of the end customer. Maximum length is 254 characters.
- **assigned_userid** – User ID of the user this session code is assigned to.
- **end_customer_link** – Link for the end customer.
- **supporter_link** – Link for the supporter.
- **custom_api** – Custom fields, this stores any arbitrary string but is only available to API functions. It is limited to 4000 characters.

Authentication

User or company access token. Scope: Sessions.ReadAll or Sessions.ReadOwn.

Description

Returns information for one session code. It will return exactly the same data that a POST to /sessions would return except that some of the fields may have changed values.

Example

Request

```
GET /api/v1/sessions/s15-542-091
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{"code" : "s15-542-091",
 "state" : "open",
 "groupid" : "g425123356",
 "waiting_message" : "",
 "description" : "HELP!!!11",
 "end_customer" : { "name" : "Max", "email" : "" },
 "assigned_userid" : "u7254190",
 "end_customer_link" : "https://get.teamviewer.com/...",
 "supporter_link" : "https://get.teamviewer.com/...",
}
```

3.6.4 PUT /api/v1/sessions/<code> (modify info for a certain session code)

Parameters

- **groupid** (*optional*) – Group ID where this session will be moved to.
- **groupname** *(optional)* – Group name where this session will be moved to. If both groupname and groupid are set, the group with the specified ID must have the specified name. Otherwise an error is returned. This parameter can only be used by applications with user-level access.
- **waiting_message** (*optional*) – Message displayed to the waiting end customer.
- **description** (*optional*) – Description for this session code.

- **end_customer** (*optional*) – End customer info
  - **name** (*optional*) – Name of the end customer.
  - **email** (*optional*) – Email of the end customer.
- **assigned_userid** (*optional*) – User ID of the user to assign this session to. Set to `u0` to unassign session.
- **custom_api** (*optional*) – Custom fields, this stores any arbitrary string but is only available to API functions. It is limited to 4000 characters.

Return values

HTTP status 204 if changes succeeded.

Authentication

User or company access token. Scope: Sessions.ModifyAll or Sessions.ModifyOwn.

Description

Modifies an existing session code.

Example

Request

```
PUT /api/v1/sessions/s13-123-123
  Content-Type: application/json

{"description" : "Still not working."}
```

Response

```
HTTP/1.1 204 No Content
```

3.7 Connection Reporting

3.7.1 GET /api/v1/reports/connections (list connection reports)

Parameters

- **username** (*optional*) – Filter by user name of the person who started the connection.
- **userid** (*optional*) – Filter by user ID of the person who started the connection.
- **groupid** (*optional*) – Filter by group ID where the target device or user was in.
- **devicename** (*optional*) – Filter by target device name.
- **deviceid** (*optional*) – Filter by device ID.
- **from_date** (*optional*) – First **start_date** for all listed connections. Parameter must contain a date and can also contain a time. If no time is specified it defaults to 00:00:00Z. Connections with **start_date** equal to **from_date** are **included** in the results.
- **to_date** (*optional*) – Last **start_date** for all listed connections. Parameter must contain a date and can also contain a time. If no time is specified it defaults to 00:00:00Z. Connections with **start_date** equal to **to_date** are **excluded** from the results.

- **offset_id** (*optional*) – All returned reports will follow the report-ID given in this field. The given report-ID is excluded from the results.
- **has_code** (*optional*) – Filters out reports that have no session code if true or that have a session code if false. If the parameter is left out both types will be returned.

Additional parameters for connections that were done with a session code:

- **session_code** (*optional*) – If specified the response will contain only connections for this session code.

Return values

- **records** – List of remote control connections
  - **id** – Report-ID
  - **userid** – User ID of the person who started the connection.
  - **username** – User name of the person who started the connection.
  - **deviceid** – Device ID (Dyngate-ID) of the target device.
  - **devicename** – Device name of the target device.
  - **groupid** – ID of the group where the device is located.
  - **groupname** – Name of the group where the device is located.
  - **start_date** – Start date and time of the connection.
  - **end_date** – End date and time of the connection.
  - **fee** – Costs for the connection.
  - **currency** – Currency for the **fee** field.
  - **billing_state** – Can be `Bill`, `Billed` or `DoNotBill`.
  - **notes** – Notes for this connection.
- **records_remaining** (*optional*)– Number of records after the ones listed here. Can be omitted if there are no more reports left.
- **next_offset** (*optional*) – ID of the last returned report in this response. Can be used as **offset_id** in a follow-up request to get the next set of connection reports.

Additional return parameters when the connection was done with a session code

- **records**
  - **assigned_userid** – User ID of the user this session code is assigned to.
  - **assigned_at** – Date when the session code was last assigned to another user.
  - **session_code** – Session code.
  - **session_created_at** – Date when the session code was created.
  - **valid_until** – Date until which the session code is/was valid.
  - **description** – Description for the session code.
  - **custom_api** – Custom fields, this stores any arbitrary string but is only available to API functions. It is limited to 4000 characters.

Authentication

User or company access token. Scope: Connections.Read.

Description

Returns a list of connection reports. The list is limited to 1000 reports per request. If there are more reports the **reports_remaining** field will tell you how many. The **next_offset** field will contain the offset ID to get the next 1000 (or less) reports and should be used as **offset_id** parameter for the next request.

If you want to get connections for a single day or multiple days, use the first day at 0:00 for the **from_date** parameter and the day after the last day at 0:00 for the **to_date** parameter.

Example

Request

```
GET /api/v1/reports/connections?username=Adam
```

Response

```
HTTP/1.1 200 OK
  Content-Type: application/json

{"records" : [
  { "id" : "B144268F-5A3A-4130-9054-A8F4598B0125",
    "username" : "Adam",
    "userid" : "u4387123",
    "devicename" : "Server 15",
    "deviceid" : "20301234",
    "start_date" : "2013-01-16T19:20:30Z",
    "end_date" : "20130116T194014Z",
    "bill" : "0.00",
    "currency" : "EUR",
    "billing_state" : "Bill",
    "notes" : "fixed webserver"
  },
  { "id" : "CAB5E30E-7A51-4F74-A9AE-089EE206D220",
    "username" : "Adam",
    "userid" : "u4387123",
    "devicename" : "Server 12",
    "deviceid" : "20301234",
    "start_date" : "20130117T144011Z",
    "end_date" : "20130117T151324Z",
    "bill" : "0.00",
    "currency" : "EUR",
    "billing_state" : "DoNotBill",
    "notes" : "installed Windows updates"
  },
  { "id" : "F846B994-4259-4F00-BEC0-258D12A6C0BE",
    "username" : "Adam",
    "userid" : "u4387123",
    "devicename" : "Server 12",
    "deviceid" : "20301234",
    "start_date" : "20130117T152004Z",
    "end_date" : "20130117T152351Z",
    "bill" : "0.00",
    "currency" : "EUR",
    "billing_state" : "DoNotBill",
    "notes" : "server rebooted and running again"
  }
]}
```

3.7.2 PUT /api/v1/reports/connections/<rID> (change connection report)

Parameters

- **billing_state** (*optional*) – Can be `Bill`, `Billed` or `DoNotBill`.
- **notes** (*optional*) – Notes for this connection.

Return values

HTTP status code 204.

Authentication

User or company access token. Scope: Connections.Modify.

Description

Changes a field in the connection report.

Example

Request

```
PUT /api/v1/reports/connections/F846B994-4259-4F00-BEC0-258D12A6C0BE
  Content-Type: application/json

{"notes" : "server rebooted but not fixed yet."}
```

Response

```
HTTP/1.1 204 No Content
```

3.7.3 DELETE /api/v1/reports/connections/<rID> (delete a connection report)

Parameters

None.

Return values

HTTP status code 200.

Authentication

User or company access token. Scope: Connections.Delete.

Description

Deletes a connection report.

Example

Request

```
DELETE /api/v1/reports/connections/F846B994-4259-4F00-BEC0-258D12A6C0BE
```

Response

```
HTTP/1.1 200 OK
```

4 Errors

4.1 HTTP response codes

200 – OK: Used for successful GET, POST (that didn't result in a new resource) and DELETE.

201 – Created: POST that resulted in a new resource.

204 – No Content: Used for PUT to indicate that the update succeeded, but no content is included in the response.

400 – Bad Request: One or more parameters for this function is either missing or unknown. Details should be included in the returned JSON.

401 – Unauthorized: Access token not valid (expired, revoked, …) or not included in the header.

403 – Forbidden / Rate Limit Reached: IP blocked or rate limit reached.

500 – Internal Server Error: Some (unexpected) error on the server. The same request should work if the server works as intended.

4.2 JSON error responses

If there is an error while processing a request, the API server returns a 4xx/5xx HTTP status code with a JSON in the body with the following parameters:

- **error** – A short string describing the category of error.
- **error_description** – A longer string containing a human readable error message.
- **error_code** – A number that is unique for each type of error.
- **error_signature** (*optional*) – A number that we can use to find the log entry if there is one. This should be unique for every time an error happens. The parameter may be omitted if there was nothing logged.

Valid values for the error field are:

- **invalid_request** – The request is missing a required parameter, includes an unsupported parameter or parameter value, repeats the same parameter, uses more than one method for including an access token, or is otherwise malformed. Should be used with HTTP response code 400.
- **invalid_token** – The access token provided is revoked, malformed, or invalid. Should be used with HTTP response code 401 (Unauthorized).
- **internal_error** – There was an error while processing the request. The error was caused by an error on our servers that should not happen and can indicate some problems at our end. Error code and signature can be used to debug the error. Should be used with HTTP status code 500 (Internal Server Error).
- **blocked** – The request was blocked. That should only happen when the IP was blocked.

- **rate_limit_reached** – Too many calls to a single function with the same access token.
- **token_expired** – Access token is expired. A new access token needs to be requested.
- **invalid_client** – Client ID was invalid.
- **email_in_use** – Returned during account creation or when changing the email if the email is already used by another account.

invalid_request and **invalid_token** are taken from http://self-issued.info/docs/draft-ietf-oauth-v2-bearer.html#resource-error-codes (with the exception that they are not included in the WWW-Authenticate header field but the returned JSON).

5 Licensing

For certain functions in the API a TeamViewer License is required. These functions are:

- User level access: connection reporting
- Company level access: all functions

These functions can only be used with a TeamViewer 9 Premium or Corporate license. All other functions are available with any license and free for private use.

Without a valid license it is not possible to create private Script Tokens or grant access to public Apps that requires any of the aforementioned functions.

To purchase or update your current license, visit http://www.teamviewer.com/licensing.

6 Contact

If you have questions or feedback, please contact support@teamviewer.com.