API Specification

CCB System Documentation

Exported on 04/15/2020

Table of Contents

Version	Description
0.8	Cleanup of unsupported planned methods
0.9	Add Board Permissions Endpoint
0.10	Refinement of Board Permissions Endpoint
0.11	Update of the General Documentation
0.12	Removal of discarded features
0.13	Add BoardData.capabilities
0.14	**BREAKING:** dropped scales part from board export urls
0.15	added Resource Owner Password Credentials Grant
0.16	added DELETE /boards{/boardid}
0.17	Board Permissions Endpoint now allows owner change of a board

Version	Breaking Change Explanation
0.14	dropped scales part from board export urls • usage analysis indicates no usage of this feature • implementation achieved enough simplification to warrant this change

1 Preamble

This document describes an API to facilitate interoperability between Conceptboard and various collaboration solutions, such as communication software, conferencing software, enterprise social software, knowledge management, and project management software.

From the perspective of such systems, Conceptboard's online whiteboard and visual collaboration eatures can represent a valuable extension to the systems' functional range.

The API aims at letting the systems' users benefit as much as possible from such additional functionality, specifically by:

- offering an interface that allows the creation of new visual workspaces ("boards") in the Conceptboard system
- offering a convenient way for users to open these boards in the browsers of their computers and mobile devices, and collaborate using the regular Conceptboard features offering end points to retrieve information about the boards

2 General Remarks

This API follows RESTful design guidelines. The API makes use of links to reference other entities. A client should be implemented with focus on the attributes and not on the concrete URIs.

2.1 Communication

The API facilitates all communication via HTTPS. API clients should accept gzip compressed responses and indicate this with the Accept-Encoding header.

2.2 Versioning Strategy

This API will follow an accretion model of versioning. If possible all changes will be additions to the system. All stated types are expandable with optional fields. New API methods can come into existence. Clients are supposed to ignore unknown fields.

2.3 Scopes

Scopes are documented with each API endpoint. A short summary of available scopes:

- `https://conceptboard.com/ns/oauth#scope-user.meta`
    - Read user information (e.g. name, settings, e-mail)
- `https://conceptboard.com/ns/oauth#scope-board.meta`
    - Read board information (e.g. name, createdat, settings)
    - Read board authorization information
- `https://conceptboard.com/ns/oauth#scope-board.meta.write`
    - Modify board information
    - Modify board authorization information
- `https://conceptboard.com/ns/oauth#scope-board.content`
    - Read board content (e.g. exports)
- `https://conceptboard.com/ns/oauth#scope-board.activity`
    - Read board activities
- `https://conceptboard.com/ns/oauth#scope-offline`
    - Request a refresh token

3 Authentication

The Interoperability API is based on sever-2-server communication. It allows a client system to interact with Conceptboard on behalf of its users, accessing a subset of their information and executing a subset of the functions that are offered via the web interface.

Authorization relies on OAuth 2.0 Authorization Framework (RFC 6749).

API Setup requires registering a specific client_id for the client system in Conceptboard as well as redirect_uri templates to strengthen the security of the OAuth Protocol Flows. The used Client Authentication mechanism might require further exchange of information.

The OAuth Endpoints support the following Client Authentication Mechanisms:

- HTTP Basic (RFC 6749)
- Client Assertion (RFC 7521) with JSON Web Tokens with JWS (RFC 7523, RFC 7518)
    - HMAC with SHA-2 Functions

The authorization endpoint supports the Authorization Code Grant user flows.

The token endpoint supports the Authorization Code Grant, Refresh Token Grant and Resource Owner Password Credentials Grant. It always requires client authentication. The response will include an access token of token_type Bearer, the expiry time and a refresh token. All access tokens will only be accepted as part of an Authorization header in further resource requests. The refresh token can be used with the Token Endpoint to get a fresh access token.

The Resource Owner Password Credentials Grant will only be made available after discussion of the intended use case and clarification of the security risks involved with this grant type.

3.1 Api Client Registration

All API Clients require Client Authentication. Public API Clients (in the understanding of the OAuth Framework) are therefore not supported. An API Client has to be setup with the following settings:

- API Clientid
- API Client Displayname (friendly name of the API Client)
- Client Authentication Credentials
- redirect_uri
    - can be specified as a regular expression pattern
- allowed scopes (default: all scopes)
- allowed origins (default: none)
    - origins for which CORS requests can be issued to API endpoints
    - CORS is not available for OAuth Authorization and Token Endpoints
- blacklisted scopes (default: none)
- allowed Conceptboard teams
    - API Clients are typically restricted to only be able to authenticate as users of a specific team
- allowed grant types (default: Authorization Code Grant and Refresh Token Grant)
    - Resource Owner Password Credentials Grant requires a restriction of the allowed Conceptboard teams and is only available upon request and if no other way can be found to support the use case

4 Endpoints

The described endpoints below list all supported attributes of the request body and the reponse attributes. The following rules apply:

- If nothing else is specified the Content Type of all responses is application/json
- Endpoints are described by HTTP_METHOD PATH
- URI patterns are described with URI TemplatesRFC6570

4.1 GET `/oauth2/authorize`

This is the Authorization Endpoint to initiate the OAuth2 Authorization Code Grant request flow.

4.1.1 Request

The Authorization Request processes the following query parameters:

`response_type`	must be equal to `code`
`client_id`	must be equal to the `client_id` established during application setup
`redirect_uri`	must be provided and match the `redirect_uri` templates established during application setup
`state`	must be provided and allow prevention of CSRF
`scope`	Space separated list of scopes requested
<client authentication related>	Additional parameters depending on the client authentication mechanism

This endpoint will require the user to log in if necessary and redirects the user to a consent screen to confirm the requested api access. If the user accepts the api access the response will be delivered to the indicated `redirect_uri` with the following additional query parameters:

`code`	the authorization code
`state`	the exact value of the `state` parameter of the Authorization Request

The authorization code will have a maximum lifetime of 10 minutes and can only be used once.

4.1.2 Response

Redirect of the clientURI to the preregistered redirectURI with the parameters code and state.

4.2 POST `/oauth2/token`

This is the Token Endpoint of the OAuth 2.0 Authorization Framework. It is used to exchange authorization code grants, refresh token grants and resource owner password credentials grants for access tokens. It requires Client Authentication.

4.2.1 Request

The token request must be `application/x-www-form-urlencoded`. The Token Request processes the following query parameters:

`grant_type`	must be equal to the string `authorization_code` or `refresh_token` or `password`
`code`	if grant_type is equal to `authorization_code`; must be equal to the `code` received from the authorization endpoint redirect
`refresh_token`	if grant_type is equal to `refresh_token`; must be equal to the last issued `refresh_token`
`scope`	optional; if grant_type is equal to `refresh_token` or `password`; restrict scopes of the returned access token. for refresh token grants the scopes must not contain more scopes then requested when the refresh_token was created.
`redirect_uri`	if grant_type is equal to `authorization_code`; must match the `redirect_uri` used for the authorization request to obtain this authorization code
`username`	if grant_type is equal to `password`
`password`	if grant_type is equal to `password`
<client authentication related>	Additional parameters depending on the client authentication mechanism

4.2.2 Response

Body: access token, refresh token, scope, token type, expiry date

4.3 GET `/boards{/boardid}` [board.meta]

This endpoint resolves the IRI of a board. It returns basic information about a board that is accessible by the user.

4.3.1 Request

Body: empty

4.3.2 Response

- If the board exists and is accessible:
  Status: 200 OK
  Body: a BoardsResponse(see page 0)

4.4 DELETE `/boards{/boardid}` [board.meta.write]

This endpoint can be used to delete a board.

4.4.1 Request

Body: empty

4.4.2 Response

successful deletion results in Status 204 No Content

4.5 GET `/boards{/boardid}/exports/…` [board.content]

This endpoint retrieves images for boards or parts of boards. Responses from other endpoints can include those links. The requesting user needs to have at least read access to the board to retrieve this image.

Construction rules for export urls:

- Start with a board IRI
  `/boards{/id}`
  => `/boards/12345`
- Add literal /exports path element
  `/boards{/id}/exports`
  => `/boards/12345/exports`
- If a specific version is requested add literal /versions path element and version path style parameters
  VERSION: the version path element specifies the version of the board, that shall be used to create the image. The board's version, as well as the path element, is made from two components the hi and

low version number; if a version is given, the response can be considered immutable.
`/boards{/id}/exports/versions/{;version*}`
`=> /boards{/id}/exports/versions/{;hi}{;low}`
`=> /boards/12345/exports/versions/;hi=1;low=12`

- If a specific size is requested add literal /sizes path element and size path style parameters
  SIZE: the size path element defines the size of the image that is requested, independently from the board size or the area of the board that is requested. All requested board content will be scaled to match the defined size. The path element is made from two components, the height and the width in pixels. In addition to a concrete value the string “orig” can be used to let the size be adapted to the area’s aspect ratio. This allows to give a concrete width and let the height be automatically determined by the content of the area. Giving a larger size will never scale the content up. If the size is larger than the content the image will have additional margins on all sides in the board background color. To actually get an upscaled version use the scale parameter.
  `/boards{/id}/exports/sizes/{;size*}`
  `=> /boards{/id}/exports/sizes/{;height}{;width}`
  `=> /boards/12345/exports/sizes/;height=orig;width=320`
- If a specific area is requested add literal /areas path element and area path style parameters
  AREA: the area path element defines the area of the board that shall be used to create the image from. It is made from four components, an x and y coordinate for the upper left corner of the area and a height and width for the dimensions of the area in internal board coordinates.
  `/boards{/id}/exports/areas/{;area*}`
  `=> /boards{/id}/exports/sizes/{;height}{;width}{;x}{;y}`
  `=> /boards/12345/exports/sizes/;height=100;width=320;x=1200;y=390`

The path elements version, size, area and scale can be combined or left out in any combination (the ordering must follow the construction rule order).

4.5.1 Request

Body: empty

4.5.2 Response

Content-Type: image/png

4.6 GET `/users/me/boards` [board.meta]

A GET request to this endpoint retrieves the list of accessible boards.

4.6.1 Request

Cursored Paging

<Cursored Paging(see page 0)>		

4.6.2 Response

Status: 200 OK

Body: a BoardsResponse with CursoredPaging support

4.7 POST `/users/me/boards` [board.meta.write]

A POST request to this endpoint will create a new board. The user invoking this endpoint will become the owner of the board.

4.7.1 Request

Body: BoardCreationCommand(see page 0)

4.7.2 Response

Status: 201 Created

Headers: Location = IRI for the created board

Body: a BoardsResponse(see page 0)

4.8 GET `/boards{/boardid}/permissions` [board.meta]

This endpoint allows access to retrieve all permissions for the board.

4.8.1 Request

Body: empty

4.8.2 Responses

Status: 200 OK

Body: a PermissionsResponse(see page 0)

4.9 POST `/boards{/boardid}/permissions` [board.meta.write]

This endpoint allows to add or update a permission. If an e-mail grantee does not have an account yet, a personal invitation will be created which grants the permission once the grantee creates his account and activates his e-mail.

This endpoint allows changing the owner of the board. This action is restricted to the current owner of the board. The new grantee must have an account and cannot be created through an invitiation.

4.9.1 Request

Body: PermissionCreateOrUpdateRequest(see page 0)

4.9.2 Response

Status: 200 OK

Body: a PermissionsResponse(see page 0)

4.10 DELETE `/boards{/boardid}/permissions{/grantee}` [board.meta.write]

This endpoint allows to remove a permission to a board for a user. The permissions of the owner cannot be removed.

4.10.1 Request

4.10.2 Response

Status: 204 NO CONTENT

5 Errors

The API responds with appropriate HTTP Status codes to any experienced errors during the request.

6 Data Types

All lowercased types are considered primitive types. The API uses IRIs as identifiers for identifiable resources. Clients should not rely on any IRI to resolve into sth. If a resource has a resolvable url it will be the `self` link of the resource.

6.1 Primitive Types

Type	Description
string	UTF-8 encoded sequence of characters
iri	an identifier for the thing in the context of the API
url	url to access with the same authentication as the API call
urltemplate	url pattern according to RFC 6570
datetime	ISO8601 formatted timestamp
integer	a number
accessmode Enum[public_link, specific]	
userrole Enum[owner,editor,reviewer,reader,meta]	
usertype Enum[email,conceptboard]	

6.2 Optional[<A>]

The value can be absent and is not required. If present it is of type A. Will also be used if fields are required only in some cases dependent on other fields.

6.3 Collection[A]

The value is an array of objects of type A. No assumptions about the order should be made.

6.4 Enum[a,b,c]

The value is one of the given strings a,b, or c.

6.5 BoardCreationCommand

Field	Type	Description
`title`	Optional[string]	The name of the board. If absent a name will be generated by the server.
`accesscontrols`	Optional[Accesscontrols]	Access control settings for the board

The `accesscontrols` field can be used to change the default role of newly added guests. It does not allow to change the access mode of the board.

6.6 BoardsResponse

Field	Type	Description
`members`	Collection[BoardData]	collection of boards
`cursor`	Optional[String]	cursor to request the next page

6.7 BoardData

Field	Type	Description
`id`	iri	the identifier of the board
`title`	string	title of the board

Field	Type	Description
`accesscontrols`	Accesscontrols	current access control settings of the board
`boardurl`	url	url to access the board at Conceptboard
`boardexport`	urltemplate	urltemplate to retrieve exports
`boardexportPreview`	url	320xorig sized image of the board
`self`	url	url to retrieve the board data
`activities`	url	url to retrieve the board activities
`liveactivities`	url	url to retrieve the boards current activities
`capabilities`	BoardCapabilities	current capabilities the invoking user has on the board

6.8 BoardCapabilities

Field	Type	Description
`canShare`	boolean	invoking user can add/remove permissions
`canView`	boolean	invoking user can be a reader on the board
`canReview`	boolean	invoking user can be a reviewer on the board
`canEdit`	boolean	invoking user can be an editor on the board
`canDelete`	boolean	invoking user can delete the board

6.9 Accesscontrols

Field	Type	Description
`mode`	accessmode	
`role`	userrole	role assigned to new users added to the board
`hasPassphrase`	boolean	if true, new users added to the board via the board url have to provide a passphrase for access
`editorsCanShare`	boolean	if true, user

6.10 Useridentifier

Field	Type	Description
`type`	Enum[conceptboard, email]	
`value`	string	Depending on the type the value will be: • conceptboard: the identifier value will be a Conceptboard userid • email: the identifier value will be an e-mail address

6.11 PermissionsResponse

Field	Type	Description
`members`	Collection[PermissionData]	A collection of permissions
`cursor`	Optional[string]	A cursor to request the next page

6.12 PermissionCreateOrUpdateRequest

Field	Type	Description
`grantee`	Useridentifier	
`role`	userrole	the role the user should get

6.13 PermissionData

Field	Type	Description
`id`	iri	id of this permission
`self`	iri	url to retrieve this permission
`grantee`	useridentifier	primary grantee identity for this permission
`granteeIdentities`	Collection[useridentifier]	Collection of all grantee identities known about the permission. If passible an email identity will be returned.
`roles`	Collection[userrole]	roles granted by this permission. The highest role is the first
`allow`	boolean	if false, this permission is indicating a deny

7 Features

Some endpoints allow specific features which, if used, add additional attributes for requests and responses.

7.1 Cursored Paging

Some resources allow paging based on a cursor.

7.1.1 Request

Field	Type	Description
`cursor`	Optional[string]	A cursor as received before.
`pagesize`	Optional[integer]	maximum number of items in the page

7.1.2 Response

The Response will have a top level field.

Field	Type	Description
`cursor`	Optional[string]	A cursor as received before.