

Video conferencing

A meeting can get a video link automatically. An **RC Admin** picks the default provider under **Settings → External Services → Video conferencing**; every new meeting then gets a link, the link moves with the meeting when it is rescheduled, and after the meeting the attendance can be read back.

Overview

Provider	Link	Moves with the meeting	Attendance	What you need
Jitsi	room name built from the meeting	not needed (a room has no schedule)	no	nothing - the public <code>meet.jit.si</code> or your own server
Zoom	scheduled meeting	yes	yes (paid plan)	a Server-to-Server OAuth app
Webex	scheduled meeting	yes	yes	a Service App and its refresh token
Google Meet	a Meet space (no calendar event)	not needed (a space has no schedule)	yes (names only)	an OAuth client and its refresh token
Microsoft Teams	online meeting of one organizer	yes	yes	an Entra app, admin consent and an application access policy

A link is made when a meeting is created and otherwise the first time somebody opens the meeting, its reminder or its calendar export (so a meeting made before a provider was set up gets its link then). A meeting with **nobody but its organizer** is a blocker in that person's calendar and gets no link until somebody else is invited.

Making a link **never** stops a meeting from being created: if the provider refuses, the meeting is saved without a link and the reason is shown. The same holds for moving and for reading attendance.

The credentials are stored in the instance's settings (like the mail settings) and are only readable by RC Admins. All calls to the providers go out from the Resourcecloud server, through the instance's proxy if one is configured.

Test status

None of the four API providers (Zoom, Webex, Google Meet, Teams) has been run against a real account yet. The code is complete (create, move, attendance) and follows each provider's documented API, but nobody has connected a real account. Treat the permission names below as the starting point from the code and the providers' documentation, and confirm them the first time you connect. Jitsi builds a link and calls nothing, so there is nothing to test.

Provider	Create a link	Move	Attendance	Run against a real account
Jitsi	by construction	n/a	n/a	not needed
Zoom	not tested	not tested	not tested	open
Webex	not tested	not tested	not tested	open
Google Meet	not tested	n/a	not tested	open
Microsoft Teams	not tested	not tested	not tested	open

If you connect one of them, a short note on what worked (and what you had to change) is very welcome - it goes into this table.

Jitsi

Choose **Jitsi** and, if you run your own server, enter its host name (just the host, e.g. `meet.example.com` - a pasted `https://` and path are removed). The default is the public `meet.jit.si`. The room name is built from the meeting; anybody with the link can join, so use your own server with authentication if the meeting is sensitive.

Zoom

- 1 In the **Zoom App Marketplace** choose **Develop → Build App → Server-to-Server OAuth App**.
- 2 Copy the **Account ID**, **Client ID** and **Client Secret** into the three Zoom fields.
- 3 Under **Scopes** add the permissions to create and to update meetings (in Zoom's current naming `meeting:write:meeting:admin` and `meeting:update:meeting:admin`; older apps call it `meeting:write:admin`) and, for attendance, `report:read:admin`. Zoom has changed these names more than once - take the ones its scope list offers for "create / update a meeting" and "view meeting reports".
- 4 Activate the app.

Meetings are created for the account's own user (`/users/me/meetings`), in the meeting timezone set under Meetings. The attendance report is a Zoom feature of paid plans and appears some time after the meeting ended.

Webex

- 1 In the **Webex Developer portal** create a **Service App** with the scope `meeting:schedules_write` (create and move meetings) and, for attendance, `meeting:participants_read`.
- 2 Ask the Webex org admin to authorize the Service App for the organization.

- 3 Copy the **Client ID**, **Client Secret** and the **Refresh Token** of the authorized app into the Webex fields.

Webex may hand out a new refresh token when the old one is used; Resourcecloud stores the new one automatically, so it stays valid. If nobody creates a meeting for a long time the token can expire - then enter a new one.

Google Meet

- 1 In the **Google Cloud console** enable the **Google Meet REST API** and create an **OAuth client**.
- 2 Authorize it once for **offline** access with the scope `https://www.googleapis.com/auth/meetings.space.created` (for attendance the Meet API's read scope for conference records may be needed as well) and keep the **refresh token**.
- 3 Copy the **Client ID**, **Client Secret** and **Refresh Token** into the Google fields. The refresh token is long-lived and is not rotated.

A Meet space has no start or end and creates no calendar event, so nothing moves when the meeting is rescheduled. The Meet API reports attendees by name, not by e-mail, so matching attendance to the invited people usually falls back to the name.

Microsoft Teams

- 1 In **Microsoft Entra ID** register an application and create a **client secret**. Copy the **Tenant ID**, **Client ID** and **secret** into the Teams fields.
- 2 Under **API permissions** add the *application* permission `OnlineMeetings.ReadWrite.All` (Microsoft Graph) and grant **admin consent**. For attendance add `OnlineMeetingArtifact.Read.All` the same way.
- 3 Create an **application access policy** that lets the app act for the organizer (Teams PowerShell: `New-CsApplicationAccessPolicy` and `Grant-CsApplicationAccessPolicy`). Without it Graph answers 403 even with consent. It can take up to about 30 minutes to apply.

4

Enter the **organizer** - the object id or the user principal name of the user the meetings are created for. All meetings of the instance belong to this one user.

If **calendar sync** (Microsoft 365) is switched on, the Outlook event creates the Teams meeting itself and moves with it; the settings above are then only used for attendance. See the calendar guide.

When something fails

- The message names the provider and, for a refused call, the HTTP code and the beginning of the provider's own answer - that answer is usually the fastest way to the cause (*invalid_client*: wrong id or secret; *403*: a permission, consent or the Teams policy is missing; *400 invalid_grant*: the refresh token is wrong or expired).
- A message that says the credentials are **not fully configured** lists which fields are empty.
- The meeting itself is always kept. Fix the setting and open the meeting again: a meeting without a link gets one then (it needs at least two people).
- Nothing is read back "too early": an empty attendance result means the provider has no report yet (Zoom and Teams need minutes after the end), not that nobody came.