AI Tinkerers Agent API

Use the Agent API to help an AI Tinkerers member or organizer with events, chapters, community operations, documents, and other capabilities available to their account.

The API is permissioned. An Agent key acts as the member who created it, and every response is limited to what that member and key may use.

Start here

- Developer page: `https://aitinkerers.org/developers/api-keys`
- Base URL: `https://aitinkerers.org/api/agents/v1`
- API discovery: `https://aitinkerers.org/api`
- OpenAPI: `https://aitinkerers.org/api/agents/v1/openapi.yaml`
- MCP: `https://aitinkerers.org/api/agents/mcp/v1`

The OpenAPI document is capability-filtered. Without authentication it describes the shared public surface. With a valid Agent key it privately includes only the additional operations that key is authorized to use. Agents should use the authenticated OpenAPI response as the source of truth for available operations and request schemas.

Authentication

Create a personal Agent key on the developer page. Agent keys begin with `sk_`.

Send the key in a header:

```
Authorization: Bearer sk_...
```

`X-API-Key` is also supported. Never put a key in a URL or request body.

Validate a key before starting a workflow:

```
curl https://aitinkerers.org/api/agents/v1/auth/validate \
  -H "Authorization: Bearer sk_..."
```

The validation response describes the key's effective roles, scopes, and available API groups. Do not infer capabilities that are absent from that response or from the authenticated OpenAPI document.

Scopes

Keys can have narrow scopes such as:

- `read`: retrieve permitted records

- `write`: create or update permitted records
- `generate`: request supported generated content
- `export`: create supported exports

Use the narrowest scopes needed. A scope permits a type of action, but it does not grant access to every chapter, event, person, or document.

Request and response shape

Read-shaped endpoints generally support `GET`. Existing clients may also use `POST` with JSON. Mutations use `POST`.

Successful responses use this envelope:

```
{
  "ok": true,
  "data": {}
}
```

Errors use this envelope:

```
{
  "ok": false,
  "error": {
    "code": "invalid_request",
    "message": "Check the request and try again."
  }
}
```

Treat returned tokens as stable API identifiers. Pass them to later get or update calls instead of guessing database IDs.

Common organizer workflows

Find a chapter or event

Use the authenticated OpenAPI document to locate chapter lookup, event search, and upcoming event operations. Search first, retain the returned token, and then use the matching detail operation.

Public worldwide event syndication is also available without a key:

```
GET https://aitinkerers.org/api/public/v1/events
```

It returns only live, indexed, public AI Tinkerers events. It does not expose attendees, private venues, RSVP details, drafts, or organizer contact data.

Work with an organizer’s events

An organizer can read or modify only the chapters, series, and events included in their management access. Finding an event in a public catalog does not imply write access to it.

Before a mutation:

1. Confirm the authenticated OpenAPI document includes the operation.
2. Retrieve the target record through an authorized detail operation.
3. Confirm the user asked for the exact state-changing action.
4. Send only fields defined by the operation schema.
5. Read the result back when practical.

Invite one person to a private VIP dinner

The Agent API supports an authorized owner or organizer inviting one named person to an inventory-managed, invitation-only VIP dinner. This includes Mixture of Experts owners acting on Mixture dinners they may manage. The production capability is feature-gated and is available to a caller only when its operations appear in the authenticated OpenAPI document or MCP tool list. Agents must not guess or probe absent routes.

The workflow has four operations: list the caller’s eligible dinners, preview one invitation, send the reviewed invitation, and read the invitation back. It uses a personal Agent key and the authenticated member is always the recorded actor and sender. Event ownership, an explicit event organizer tag, or an invitation-manager dinner-team role is required. A public event listing, an unrelated organizer role, administrator status by itself, or possession of a private event token is not sufficient.

The approved operation names and routes are:

- `dinner_invitations_events_list`: `GET|POST /api/agents/v1/dinner_invitations/events/list`
- `dinner_invitations_preview`: `POST /api/agents/v1/dinner_invitations/preview`
- `dinner_invitations_send`: `POST /api/agents/v1/dinner_invitations/send`
- `dinner_invitations_get`: `GET|POST /api/agents/v1/dinner_invitations/get`

The recipient may be selected by an existing Client token, by an authorized Guest Radar person with an exact email, or simply by entering an email address and optional name. Existing email addresses are matched case-insensitively and reuse the existing Client. A new email creates only a minimal email identity for this invitation. It does not create an RSVP or grant attendee access at send time.

Preview shows the exact recipient, message, reply deadline, and seat effect. It holds no seat, sends nothing, and returns a signed proposal valid for 15 minutes. The short lifetime prevents an agent from sending old reviewed copy or relying on stale capacity. If it expires, preview again. It has no effect on an invitation that was already sent.

Send uses an idempotency key and rechecks authorization, recipient suppression, duplicates, and capacity. A successful send holds exactly one seat immediately. When the recipient verifies the invited email and accepts, that same seat moves from Held to Occupied and the RSVP is created. Acceptance never consumes a second seat. Decline or expiry releases the hold once.

Capacity has two limits. Every dinner enforces its global hard capacity, occupied seats, active holds, and protected buffer. For Mixture of Experts, the lead organizer uses the open-seat pool: every allocatable guest seat not assigned to a collaborator. Collaborators require explicit fixed invite limits. An ordinary private VIP dinner uses the shared pool if no limits exist; after any collaborator limit is configured, other inviters need their own limit. Series or event ownership authorizes the tool but never bypasses the hard dinner capacity.

`409 event_capacity_exhausted` includes overall capacity, occupied, held, protected, and available counts. `409 organizer_allocation_exhausted` also includes the caller's seat limit, used count, and remaining count. Both errors make no changes and give a corrective action. The agent may ask an event owner to allocate or transfer seats in the website Invitations workspace, release an unaccepted hold, or increase the event capacity as indicated. Version 1 does not expose allocation transfer through the API.

If invitation creation succeeds but delivery fails, the API returns `502 delivery_failed` with the invitation token and explicit `invitation_created` and `seat_held` flags. The agent must not say the email was sent. Retrying the same idempotency key does not create another seat; use the website Invitations workspace to review the failed delivery.

Creating the email identity, sending the invitation, and accepting it do not subscribe the person to a newsletter, chapter list, Mixture list, or general AI Tinkerers mailing list. Existing preferences are preserved. Event invitation, confirmation, reminder, venue-update, and cancellation messages are transactional messages tied only to that dinner. A new person is prompted to complete their profile after acceptance.

Work with community records

Person, RSVP, subscriber, sponsor, and related records are privacy-scoped. Responses omit protected fields the key cannot access. An omitted field should be treated as unavailable, not as evidence that the field exists with an empty value.

Do not use this API to assemble lead lists, scrape contact data, or infer hidden fields.

Read AI Tinkerers documents

Use the document search and get operations exposed by the authenticated OpenAPI document. Search results and retrieval are filtered to documents the member may access.

Use MCP

OAuth-capable MCP clients can connect to:

```
https://aitinkerers.org/api/agents/mcp/v1
```

The server supports interactive OAuth and Agent bearer keys. Its tool list is capability-filtered for the authenticated member. Refresh the tool list after permissions or scopes change.

Agent safety rules

- Never probe undocumented paths to discover unavailable products or operations.
- Never claim a write succeeded until the API returns success and, when possible, a readback confirms the new state.
- Do not turn an invitation request into an RSVP, registration, send, or other different action without explicit confirmation.
- Do not reveal private API responses outside the user's requested workflow.
- Preserve idempotency keys and concurrency values when an operation requires them.
- If an error is safe but nonspecific, explain the user action that is blocked without speculating about hidden permissions or product features.

Errors and troubleshooting

- `401`: the key is missing, invalid, expired, or revoked. Validate it and create a new key if needed.
- `403`: the requested action is not available to this member, key, or resource. Check the authenticated OpenAPI document, key scopes, and ownership of the target resource.
- `404`: the resource is unavailable or not visible to this caller. Do not use the response to infer that a hidden record exists.
- `409`: the request conflicts with current state. Refresh the record and follow any returned concurrency or idempotency guidance.
- `422`: the request shape or values are invalid. Compare the request with the authenticated OpenAPI schema.
- `429`: wait for the requested retry interval before trying again.

For persistent access problems, use the developer page to inspect or replace the key. Report the operation, HTTP status, safe error code, and request identifier. Never include the full key.