MCP Server & Claude Desktop Integration

Using the Gathering Organiser MCP connector with Claude Desktop or Claude.ai to manage gatherings conversationally.

What Is the MCP Connector?

The Gathering Organiser MCP Server lets you manage gatherings through conversational AI. Instead of clicking through the admin panel, you can ask Claude to check registrations, update checklists, draft communications, and more; all through natural language.

How it works

MCP stands for Model Context Protocol. It's an open standard that lets AI assistants (like Claude) interact with external tools and services. The Gathering Organiser MCP Server wraps the Riverland Faeries REST API into 25 tools that Claude can call on your behalf.

The flow looks like this:

1. You ask Claude something like "How many people have registered for Beltane?"
2. Claude calls the `registration_stats` tool via the MCP server
3. The MCP server queries the Riverland Faeries API
4. The results come back to Claude, who presents them in a readable format

What you can do with it

The MCP server covers seven areas of gathering management:

- **Gathering management**: create, update, and review gatherings
- **Registrations**: list registrations, update statuses, export data, check stats
- **Budget and finance**: generate budget calculations and bilingual financial reports
- **Carpooling and logistics**: view carpool matches, update match statuses, check pickup runs
- **Communications**: draft emails using bilingual templates (English/Dutch)
- **Checklists**: check planning progress, update checklist items, view overdue tasks
- **Wiki pages**: read and update gathering wiki content

Two ways to connect

The MCP server supports two transport modes:

Mode	Best for	Authentication
stdio	Local development; running the server on your own machine	Static API token

Mode	Best for	Authentication
HTTP	Production use; connecting to the hosted server at `mcp-gatherings.junovy.com`	Keycloak OAuth 2.0 (signs in with your Junovy Account)

For most organisers, the **HTTP mode** is the simplest option; you just point Claude at the hosted server and sign in with your Junovy Account. The **stdio mode** is for developers who want to run the MCP server locally.

Who can use it

You need to be a member of the organiser team for a gathering to manage it via the MCP server. The same permissions that apply in the admin panel apply when using Claude; if you can't do something in the UI, you can't do it via MCP either.

Setting Up Claude Desktop

Claude Desktop is Anthropic's desktop application for macOS and Windows. You can connect it to the Gathering Organiser MCP Server so that Claude can manage gatherings directly from your conversations.

Option 1: HTTP mode (recommended)

This connects Claude Desktop to the hosted MCP server. No local setup required.

1. Open Claude Desktop
2. Go to **Settings** (click your profile icon, then **Settings**)
3. Click **Developer** in the sidebar, then **Edit Config**
4. This opens the configuration file. Add the following to the `mcpServers` section:

```
{
  "mcpServers": {
    "gathering-organiser": {
      "url": "https://mcp-gatherings.junovy.com/mcp",
      "transport": "streamable-http"
    }
  }
}
```

5. Save the file and restart Claude Desktop
6. When you first use a gathering tool, you'll be prompted to sign in with your Junovy Account via Keycloak

Option 2: stdio mode (local development)

This runs the MCP server on your own machine. Useful for developers working on the Riverland Faeries codebase.

Prerequisites:

- Node.js 22+ (via Volta)
- pnpm 10+
- The `tenant-riverland-faeries` repository cloned locally

Steps:

1. Build the MCP server:

```
cd tenant-riverland-faeries/apps/mcp
pnpm install
pnpm build
```

2. Open your Claude Desktop config file and add:

```
{
  "mcpServers": {
    "gathering-organiser": {
      "command": "node",
      "args": ["/path/to/tenant-riverland-faeries/apps/mcp/dist/index.js"],
      "env": {
        "GATHERING_API_URL": "http://localhost:3000",
        "GATHERING_API_TOKEN": "your-api-token"
      }
    }
  }
}
```

Replace `/path/to/` with the actual path to your local repository, and `your-api-token` with a valid API token.

3. Save and restart Claude Desktop

Config file locations

Platform	Path
macOS	`~/Library/Application Support/Claude/claude_desktop_config.json`
Windows	`%APPDATA%\Claude\claude_desktop_config.json`

Verifying the connection

After restarting Claude Desktop, you should see a hammer icon (□□) in the input area. Click it to see the list of available gathering tools. If the tools appear, the connection is working.

Try asking Claude: "List all gatherings" to confirm everything is set up correctly.

Troubleshooting

- **Tools don't appear**: check that the config JSON is valid (no trailing commas, correct brackets) and restart Claude Desktop
- **Authentication errors (HTTP mode)**: make sure you can sign in to `riverland.faeries.eu` with your Junovy Account; the MCP server uses the same Keycloak authentication
- **Connection refused (stdio mode)**: verify the path in `args` points to the built `dist/index.js` file, and that the API URL and token are correct
- **Server not responding**: check that the Riverland Faeries API is running (locally for stdio, or that `mcp-gatherings.junovy.com` is reachable for HTTP)

Setting Up Claude.ai (Cowork Mode)

If you use Claude through the web at claude.ai or the Claude desktop app's Cowork mode, you can add the Gathering Organiser MCP Server as a remote connector. This works with the hosted HTTP server, so no local installation is needed.

Adding the connector

1. Open Claude.ai or the Claude desktop app
2. Navigate to **Settings > Connectors** (or **Integrations**, depending on your version)
3. Click **Add Connector** or **Add MCP Server**
4. Enter the following details:

Field	Value
Name	Gathering Organiser
URL	`https://mcp-gatherings.junovy.com/mcp`
Transport	Streamable HTTP

5. Click **Save** or **Connect**
6. You'll be redirected to sign in with your Junovy Account via Keycloak
7. After signing in, the connector is active and Claude can use the gathering tools

Using the connector

Once connected, you can ask Claude about your gatherings in any conversation. For example:

- "Show me all published gatherings"
- "How many registrations does Beltane 2026 have?"
- "What's the checklist progress for the upcoming gathering?"

Claude will use the appropriate MCP tool to fetch or update the information.

Managing the connector

- To disconnect, go back to **Settings > Connectors** and remove the Gathering Organiser entry
- If your session expires, Claude will prompt you to re-authenticate

- The connector uses OAuth 2.0 with PKCE, so your credentials are never stored by the MCP server; authentication is handled entirely through Keycloak

Tips

- The connector respects the same permissions as the admin panel; you can only manage gatherings you're an organiser for
- If you're using Cowork mode in the Claude desktop app, the connector works the same way as on claude.ai
- You can have the connector active alongside other MCP connectors without any conflicts

Available MCP Tools

The Gathering Organiser MCP Server exposes 25 tools across seven categories. This page lists all of them with a brief description of what each one does.

Gathering management

Tool	Description
`gathering_list`	List all gatherings, optionally filtered by status or type
`gathering_get`	Get full details of a specific gathering
`gathering_create`	Create a new gathering
`gathering_update`	Update an existing gathering (title, dates, status, capacity, etc.)
`gathering_get_summary`	Get an overview including registration counts, checklist progress, and logistics status

Registration and participants

Tool	Description
`registration_list`	List registrations for a gathering, optionally filtered by status
`registration_get`	Get full details of a single registration including form responses
`registration_update_status`	Approve, waitlist, decline, or cancel a registration
`registration_stats`	Get registration statistics: totals by status, capacity usage, dietary breakdown
`registration_export`	Export registrations as CSV, JSON, markdown, or dietary summary

Budget and finance

Tool	Description
`budget_create`	Generate a budget calculation with sliding-scale pricing tiers
`budget_generate_report`	Produce a formatted bilingual (English/Dutch) budget report for sharing

Carpooling and logistics

Tool	Description
`carpool_list`	List all carpool matches for a gathering
`carpool_match`	Update a carpool match status (confirm, cancel, send introduction)
`carpool_stats`	Get carpool statistics: matches, confirmed/pending/cancelled, unmatched passengers
`pickup_list`	List all pickup runs with drivers, times, locations, and passengers

Communication and templates

Tool	Description
`comms_draft`	Draft an email using a bilingual template with gathering-specific data
`comms_list_templates`	List all available email templates
`comms_get_template`	Get details of a specific template including required parameters

Available email templates: `save-the-date`, `invitation`, `practical-update`, `thank-you`, `financial-summary`.

Planning checklists

Tool	Description
`checklist_get`	Get checklist items, optionally filtered by planning phase
`checklist_update_item`	Update a checklist item (complete, assign, add notes, set due date)
`checklist_progress`	Get overall and per-phase completion rates with overdue items

Wiki pages

Tool	Description
`wiki_get`	List all wiki pages or get a specific page's content
`wiki_update_section`	Update a wiki page's title, content, published status, or order

Tool	Description
`wiki_publish`	Publish a wiki page to make it publicly visible

Read-only vs. write tools

Most tools are read-only (they fetch data without changing anything). The tools that can make changes are: `gathering_create`, `gathering_update`, `registration_update_status`, `carpool_match`, `checklist_update_item`, `wiki_update_section`, `wiki_publish`, `budget_create`, `budget_generate_report`, and `comms_draft`.

When Claude uses a write tool, it will typically confirm the action with you first before proceeding.

Example Workflows

Here are some practical examples of how you can use the MCP connector with Claude to manage your gatherings. These show the kinds of questions you can ask and what Claude will do behind the scenes.

Checking registration status

You: "How are registrations looking for Beltane 2026?"

Claude uses `registration_stats` to pull the numbers, then presents a summary like:

- 28 approved, 5 pending, 3 waitlisted
- Capacity: 28 / 35 optimal, 28 / 45 maximum
- Dietary breakdown: 12 omnivore, 8 vegetarian, 5 vegan, 3 gluten-free

Reviewing and approving registrations

You: "Show me the pending registrations and approve the ones that look good."

Claude uses `registration_list` with a status filter, shows you each pending registration with their form responses, and then uses `registration_update_status` to approve the ones you confirm.

Checking planning progress

You: "What's left to do before the gathering?"

Claude uses `checklist_progress` to get completion rates and highlights overdue items. It might respond with something like: "Overall 72% complete. The 'one week before' phase has 3 overdue items: book firewood delivery, confirm kitchen rota, and print emergency contacts."

Coordinating carpools

You: "Are there any unmatched passengers for the gathering?"

Claude uses `carpool_stats` to check, then `carpool_list` to see the details. It can identify who still needs a ride and suggest potential matches based on location.

Drafting a save-the-date email

You: "Draft a save-the-date email for Beltane 2026."

Claude uses `comms_draft` with the `save-the-date` template. It pulls the gathering details and produces a bilingual (English/Dutch) email you can review and send.

Updating a wiki page

You: "Update the 'What to Bring' wiki page to mention that we'll have a sauna this time."

Claude uses `wiki_get` to fetch the current page content, then `wiki_update_section` to add the new information. You can review the updated content before it goes live.

Getting a gathering summary

You: "Give me a full overview of where we are with Beltane planning."

Claude uses `gathering_get_summary` to pull together registration counts, checklist progress, carpool status, and wiki page status into a single overview.

Generating a budget report

You: "Create a budget report for 35 participants over 5 days."

Claude uses `budget_generate_report` to calculate costs with sliding-scale tiers and produces a formatted bilingual report you can share with participants.

Tips for getting the best results

- Be specific about which gathering you mean if you have more than one active
- You can chain requests: "Check the registrations, then draft a practical update email with the current numbers"
- Claude will ask for confirmation before making changes (approving registrations, updating wiki pages, etc.)
- If you're not sure what tools are available, just ask Claude: "What can you help me with for gathering planning?"