Package: jirar (via r-universe)

October 31, 2026

Title Modern R Client for the Jira REST API

Version 0.1.0

Description A modern, tidy interface to the Jira REST API for both Jira Cloud and Jira Server / Data Center. Authenticate once, query issues with the Jira Query Language (JQL), and retrieve projects, fields, dashboards and more as tibbles. Built on 'httr2' with automatic pagination, informative errors and support for API tokens and personal access tokens.

License MIT + file LICENSE

URL https://github.com/JanWein/jirar

BugReports https://github.com/JanWein/jirar/issues

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Suggests dplyr, jsonlite, knitr, rmarkdown, spelling, testthat (>= 3.0.0), withr

Config/testthat/edition 3

Imports cli, httr2 (>= 1.0.0), rlang, tibble, vctrs

Depends R (>= 4.1)

Config/pak/sysreqs libssl-dev

Repository https://janwein.r-universe.dev

Date/Publication 2026-06-17 08:37:12 UTC

RemoteUrl https://github.com/janwein/jirar

RemoteRef HEAD

RemoteSha f92159cca7e136dc5f109fa7e0c97f4587b9f0df

Contents

jira_connect 2
jira_connection 3
jira_count_issues 5
jira_dashboards 5
jira_default_fields 6
jira_fields 7
jira_get_connection 7
jira_groups 8
jira_issues 9
jira_my_permissions 10
jira_myself 11
jira_permissions 12
jira_projects 12
jira_server_info 13

Index **15**

`jira_connect` *Connect to Jira and register the default connection*

Description

`jira_connect()` creates a `jira_connection()`, verifies it by requesting the current user (unless `check = FALSE`), and stores it as the session default so other `jira_*()` functions can be called without passing `connection` explicitly. This is the modern replacement for `JirAgileR`'s `save_jira_credentials()`.

Usage

```
jira_connect(
  url = NULL,
  user = NULL,
  token = NULL,
  deployment = c("auto", "cloud", "server"),
  check = TRUE,
  set_default = TRUE
)
```

Arguments

`url`	Base URL of the Jira instance, e.g. `"https://your-domain.atlassian.net"`. Defaults to the `JIRA_URL` environment variable.
`user`	Account email (Cloud) or username (Server/Data Center basic auth). Leave `NULL` for Server/Data Center Personal Access Token auth. Defaults to the `JIRA_USER` environment variable.
`token`	API token (Cloud), Personal Access Token, or password (Server basic auth). Defaults to the `JIRA_TOKEN` environment variable.

deployment	One of "auto", "cloud" or "server". With "auto" (the default), the deployment is inferred from url (a host ending in .atlassian.net is treated as Cloud, everything else as Server / Data Center). The deployment selects the REST API version (v3 for Cloud, v2 for Server) and the issue-search pagination strategy.
check	If TRUE (the default), verify the connection by calling jira_myself(). A failure here usually means the URL or credentials are wrong.
set_default	If TRUE (the default), register the connection as the session default (see jira_get_connection()).

Value

The jira_connection, invisibly.

Examples

```
## Not run:
# Reads JIRA_URL, JIRA_USER and JIRA_TOKEN from the environment:
jira_connect()

# Or pass values directly:
jira_connect(
  url = "https://example.atlassian.net",
  user = "me@example.com",
  token = Sys.getenv("MY_TOKEN")
)

## End(Not run)
```

---

jira_connection *Create a connection to a Jira instance*

Description

jira_connection() builds (but does not verify) a connection object that describes how to reach a Jira instance and how to authenticate. Most users should call jira_connect() instead, which additionally verifies the credentials and registers the connection as the session default.

Usage

```
jira_connection(
  url = NULL,
  user = NULL,
  token = NULL,
  deployment = c("auto", "cloud", "server"),
  auth = c("auto", "basic", "bearer"),
  api_version = NULL
)
```

Arguments

`url`	Base URL of the Jira instance, e.g. `"https://your-domain.atlassian.net"`. Defaults to the `JIRA_URL` environment variable.
`user`	Account email (Cloud) or username (Server/Data Center basic auth). Leave `NULL` for Server/Data Center Personal Access Token auth. Defaults to the `JIRA_USER` environment variable.
`token`	API token (Cloud), Personal Access Token, or password (Server basic auth). Defaults to the `JIRA_TOKEN` environment variable.
`deployment`	One of `"auto"`, `"cloud"` or `"server"`. With `"auto"` (the default), the deployment is inferred from `url` (a host ending in `.atlassian.net` is treated as Cloud, everything else as Server / Data Center). The deployment selects the REST API version (v3 for Cloud, v2 for Server) and the issue-search pagination strategy.
`auth`	One of `"auto"`, `"basic"` or `"bearer"`. With `"auto"`, basic auth is used for Cloud and for Server when `user` is supplied; bearer (PAT) auth is used for Server when `user` is empty.
`api_version`	REST API version as a string. Defaults to `"3"` for Cloud and `"2"` for Server / Data Center.

Details

Authentication differs by deployment:

- **Jira Cloud** (`*.atlassian.net`) uses HTTP basic authentication with your account email as `user` and an API token as `token`. Create a token at https://id.atlassian.com/manage-profile/security/api-tokens.
- **Jira Server / Data Center** uses a Personal Access Token sent as a bearer token (supply `token` and leave `user` empty), or basic authentication with a username and password (supply both `user` and `token`).

Any argument left as `NULL` falls back to an environment variable: `JIRA_URL`, `JIRA_USER` and `JIRA_TOKEN`.

Value

A `jira_connection` object.

Examples

```
conn <- jira_connection(
  url = "https://example.atlassian.net",
  user = "me@example.com",
  token = "secret-api-token"
)
conn
```

jira_count_issues *Count the issues matching a JQL query*

Description

Returns the number of issues matching a JQL query without retrieving them. On Jira Cloud this uses the approximate-count endpoint (the result is an estimate for large result sets); on Jira Server / Data Center it returns the exact `total`.

Usage

```
jira_count_issues(jql = NULL, connection = jira_get_connection())
```

Arguments

`jql`	A JQL query string, e.g. `"project = ABC AND status = Open"`. Pass `NULL` (the default) or `""` to retrieve **all** issues visible to the authenticated user. Beware: on a large instance this can be a lot of issues, so consider pairing it with `max_results`.
`connection`	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Value

A single integer count.

Examples

```
## Not run:
jira_count_issues("project = ABC AND created >= -7d")

## End(Not run)
```

jira_dashboards *Retrieve Jira dashboards*

Description

Returns the dashboards visible to the authenticated user as a tibble. Pagination is handled automatically.

Usage

```
jira_dashboards(
  filter = NULL,
  max_results = Inf,
  connection = jira_get_connection()
)
```

Arguments

filter	Optional filter: "my" for dashboards you own or "favourite" for your favourites. NULL (the default) returns all visible dashboards.
max_results	Maximum number of dashboards to return. Defaults to Inf.
connection	A jira_connection. Defaults to the session connection registered with jira_connect().

Value

A tibble of dashboards, one row per dashboard.

Examples

```
## Not run:
jira_connect()
jira_dashboards()
jira_dashboards(filter = "favourite")

## End(Not run)
```

---

jira_default_fields *Default and supported JQL fields*

Description

jira_default_fields() returns the fields requested by jira_issues() when none are supplied: a compact, broadly useful set. jira_supported_fields() returns the set of well-known system fields that flatten cleanly into columns, mirroring the fields supported by the JirAgileR package.

Usage

```
jira_default_fields()

jira_supported_fields()
```

Value

A character vector of field names.

Examples

```
jira_default_fields()
jira_supported_fields()
```

jira_fields *Retrieve the fields defined on a Jira instance*

Description

Returns all system and custom fields, including the mapping between custom field ids (e.g. `customfield_10010`) and their human-readable names and JQL clause names. Useful for discovering field ids to pass to `jira_issues()`.

Usage

```
jira_fields(connection = jira_get_connection())
```

Arguments

`connection`	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Value

A tibble of fields, one row per field.

Examples

```
## Not run:
jira_connect()
jira_fields()

## End(Not run)
```

jira_get_connection *Get or set the default Jira connection*

Description

`jira_get_connection()` returns the connection registered by `jira_connect()`; it errors if none has been set. `jira_set_connection()` registers a connection as the session default. These power the `connection = jira_get_connection()` default of the other `jira_*()` functions.

Usage

```
jira_get_connection()

jira_set_connection(connection)
```

Arguments

`connection`	A `jira_connection` created by `jira_connection()` or `jira_connect()`.

Value

`jira_get_connection()` returns a `jira_connection`. `jira_set_connection()` returns the previous default invisibly.

`jira_groups` *Retrieve Jira groups*

Description

Returns Jira groups as a tibble. The endpoint used depends on the deployment and whether `query` is supplied:

Usage

```
jira_groups(
  query = NULL,
  max_results = Inf,
  connection = jira_get_connection()
)
```

Arguments

`query`	Optional substring to search group names for. Note that on Cloud, passing `query` (even `""`) switches from the unlimited bulk listing to the capped picker search.
`max_results`	Maximum number of groups to return. Defaults to `Inf`. For picker searches the result is additionally capped at 1000 by the API.
`connection`	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Details

- **Jira Cloud,** `query` = `NULL`: all groups are listed via the paginated bulk endpoint (`group/bulk`), honouring `max_results`.
- **Jira Cloud with a** `query`**, or any Jira Server / Data Center call**: a name search is performed via the group picker (`groups/picker`), which the API caps at 1000 results.

Value

A tibble of groups, one row per group.

Examples

```
## Not run:
jira_connect()
jira_groups()
jira_groups(query = "admin")

## End(Not run)
```

jira_issues *Retrieve Jira issues with a JQL query*

Description

Runs a Jira Query Language (JQL) query and returns the matching issues as a tibble, one row per issue. Pagination is handled automatically: token-based (`nextPageToken`) on Jira Cloud and offset-based (`startAt`) on Jira Server / Data Center.

Usage

```
jira_issues(
  jql = NULL,
  fields = jira_default_fields(),
  expand = NULL,
  max_results = Inf,
  page_size = NULL,
  parse = TRUE,
  connection = jira_get_connection()
)
```

Arguments

`jql`	A JQL query string, e.g. `"project = ABC AND status = Open"`. Pass `NULL` (the default) or `""` to retrieve **all** issues visible to the authenticated user. Beware: on a large instance this can be a lot of issues, so consider pairing it with `max_results`.
`fields`	Character vector of fields to retrieve. Use specific field ids or names (e.g. `c("summary", "status", "assignee")`), or the sentinels `"*all"` / `"*navigable"`. Defaults to `jira_default_fields()`.
`expand`	Optional character vector of entities to expand, e.g. `"changelog"` or `"renderedFields"`.
`max_results`	Maximum number of issues to return across all pages. Defaults to `Inf` (all matching issues). Use a finite value to cap large queries.
`page_size`	Issues requested per page (the Jira `maxResults` parameter). Defaults to 100 on Cloud and 50 on Server / Data Center. The server may return fewer; jirar keeps paging regardless.
`parse`	If `TRUE` (the default), return a flattened tibble. If `FALSE`, return the raw list of issue objects as parsed from JSON.
`connection`	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Details

On Jira Cloud the search endpoint only returns the issue `id` and `key` unless `fields` are requested explicitly, so always pass the fields you need (or the sentinels `"*all"` / `"*navigable"`). Nested fields are flattened to snake_case columns (e.g. `status_name`, `assignee_displayname`, `components_name`), datetimes are parsed to `POSIXct`, and rich-text (Atlassian Document Format) bodies such as `description` are converted to plain text.

Note that array fields (e.g. `components`, `labels`, `fixVersions`) are collapsed to comma-separated strings, which loses the association between an object's sub-fields. For structured access to such fields, use `parse = FALSE` and work with the raw issue list.

Value

A tibble with one row per issue. Always includes `issue_id` and `key`; the remaining columns depend on `fields` and are named after the (possibly nested) field, e.g. `summary`, `status_name`, `assignee_displayname`. Returns the raw list when `parse = FALSE`.

Examples

```
## Not run:
jira_connect()

# All issues you can see (no JQL needed)
jira_issues()
jira_issues(max_results = 200)

# Or filter with JQL
jira_issues("project = ABC AND statusCategory != Done")
jira_issues(
  "assignee = currentUser() ORDER BY updated DESC",
  fields = c("summary", "status", "updated"),
  max_results = 100
)

## End(Not run)
```

---

jira_my_permissions *Check the current user's permissions*

Description

Returns, for each requested permission, whether the authenticated user holds it (optionally within a project or issue context). On Jira Cloud the `permissions` argument is required by the API; if omitted, jirar fills it with the full permission catalogue from `jira_permissions()`.

Usage

```
jira_my_permissions(
  permissions = NULL,
  project_key = NULL,
  issue_key = NULL,
  connection = jira_get_connection()
)
```

Arguments

permissions	Character vector of permission keys to check, e.g. c("BROWSE_PROJECTS", "EDIT_ISSUES"). Required on Jira Cloud: if NULL, jirar makes an extra call to jira_permissions() to fetch the full catalogue, so passing the specific keys you need is more efficient. On Jira Server / Data Center this is not auto-filled.
project_key	Optional project key to scope the check to. Mutually exclusive with issue_key.
issue_key	Optional issue key to scope the check to. Mutually exclusive with project_key.
connection	A jira_connection. Defaults to the session connection registered with jira_connect().

Value

A tibble of permissions with a havepermission column.

Examples

```
## Not run:
jira_connect()
jira_my_permissions(c("BROWSE_PROJECTS", "CREATE_ISSUES"))
jira_my_permissions("EDIT_ISSUES", project_key = "ABC")

## End(Not run)
```

---

jira_myself *Get the currently authenticated user*

Description

Returns the account that the connection authenticates as. This is the lightest possible call and is used by jira_connect() to verify credentials. On Jira Cloud the user is identified by accountid; on Server / Data Center by name (username).

Usage

```
jira_myself(connection = jira_get_connection())
```

Arguments

connection	A jira_connection. Defaults to the session connection registered with jira_connect().

Value

A one-row tibble describing the current user.

Examples

```
## Not run:
jira_connect()
jira_myself()

## End(Not run)
```

`jira_permissions` *Retrieve all Jira permissions*

Description

Returns the catalogue of permissions the Jira instance knows about (global and project permissions), one row per permission.

Usage

```
jira_permissions(connection = jira_get_connection())
```

Arguments

`connection`	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Value

A tibble of permissions.

Examples

```
## Not run:
jira_connect()
jira_permissions()

## End(Not run)
```

`jira_projects` *Retrieve Jira projects*

Description

Returns the projects visible to the authenticated user as a tibble. On Jira Cloud this uses the paginated `project/search` endpoint; on Jira Server / Data Center it uses `project` (a non-paginated array).

Usage

```
jira_projects(
  query = NULL,
  max_results = Inf,
  connection = jira_get_connection()
)
```

Arguments

query	Optional case-insensitive filter matched against project key and name. On Jira Cloud this is sent to the server (`project/search`); on Jira Server / Data Center, where the endpoint returns all projects at once, the filtering is applied client-side.
max_results	Maximum number of projects to return. Defaults to `Inf`.
connection	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Value

A tibble of projects, one row per project.

Examples

```
## Not run:
jira_connect()
jira_projects()
jira_projects(query = "platform")

## End(Not run)
```

---

jira_server_info *Get information about the Jira instance*

Description

Returns instance metadata, including `deploymenttype` (`Cloud`, `Server` or `DataCenter`), version and base URL.

Usage

```
jira_server_info(connection = jira_get_connection())
```

Arguments

connection	A `jira_connection`. Defaults to the session connection registered with `jira_connect()`.

Value

A one-row tibble of server information.

Examples

```
## Not run:
jira_connect()
jira_server_info()

## End(Not run)
```

Index

jira_connect, 2
jira_connect(), *3*, *5–9*, *11–13*
jira_connection, 3
jira_connection(), *2*, *7*
jira_count_issues, 5
jira_dashboards, 5
jira_default_fields, 6
jira_default_fields(), *9*
jira_fields, 7
jira_get_connection, 7
jira_get_connection(), *3*
jira_groups, 8
jira_issues, 9
jira_issues(), *6, 7*
jira_my_permissions, 10
jira_myself, 11
jira_myself(), *3*
jira_permissions, 12
jira_permissions(), *10, 11*
jira_projects, 12
jira_server_info, 13
jira_set_connection
        *(*jira_get_connection*)*, 7
jira_supported_fields
        *(*jira_default_fields*)*, 6