Package: RD1 (via r-universe)

June 3, 2026

Title DBI Client and API Wrapper for Cloudflare D1

Version 0.0.0.9000

Description A DBI-compliant database interface and API client for Cloudflare D1's serverless SQLite database. Provides wrappers for D1 REST API endpoints, including time travel and bookmark-based restoration, plus convenience functions for common multi-step workflows such as exporting and downloading databases.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Imports cli, DBI, httr2, methods, rlang

Suggests DBItest, dbplyr, dplyr, RSQLite, rstudioapi, testthat, withr

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 8.0.0

Config/pak/sysreqs libssl-dev

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

Date/Publication 2026-06-02 14:26:12 UTC

RemoteUrl https://github.com/noamross/RD1

RemoteRef HEAD

RemoteSha 477cb2664e79a050eec222e2c35f58bee9577cd9

Contents

`credentials` *Cloudflare credentials*

Description

Resolve the API token and account ID used for D1 requests. Each looks first at its argument, then an option, then an environment variable.

Usage

```
d1_token(token = NULL)

d1_account(account_id = NULL)
```

Arguments

`token`	A Cloudflare API token. Defaults to the option `RD1.token` or the environment variable `CLOUDFLARE_API_TOKEN`.
`account_id`	A Cloudflare account ID. Defaults to the option `RD1.account_id` or the environment variable `CLOUDFLARE_ACCOUNT_ID`.

Value

A length-one character string.

`d1` *RD1 DBI driver*

Description

Returns a driver object used to connect to a Cloudflare D1 database with `DBI::dbConnect()`. D1 is serverless SQLite accessed over HTTP, so a "connection" simply bundles the database ID and credentials; there is no persistent socket. `d1()`, `D1()`, `Rd1()`, and `RD1()` are interchangeable aliases.

Usage

```
d1()

## S4 method for signature 'D1Driver'
dbConnect(drv, database_id, account_id = d1_account(), token = d1_token(), ...)
```

Arguments

drv	A D1Driver, from d1().
database_id	Database UUID or name to connect to (names are resolved with d1_database_id()).
account_id, token	Cloudflare credentials. See d1_token().
...	Unused.

Value

d1() returns a D1Driver. dbConnect() returns a D1Connection.

Differences between D1 and SQLite

D1 *is* SQLite, but reached over Cloudflare's HTTP API rather than a local file. This shapes how RD1 behaves compared with a file-based driver such as RSQLite:

- **No persistent connection.** A D1Connection only bundles the database ID and credentials. DBI::dbDisconnect() is a no-op that marks the connection invalid; there is no socket to close.
- **No transactions.** Each statement is sent and committed on its own. dbBegin()/dbCommit()/dbRollback() are not supported. Atomic restores are instead available through time travel (see below).
- **Eager results.** D1 returns a query's full result in one response, so results are materialised in memory and DBI::dbFetch() simply pages through them locally.
- **Storage classes versus JSON.** Results arrive as JSON, which cannot distinguish SQLite storage classes for whole numbers (a REAL 6.0 and an INTEGER 6 both serialise as 6). For ad-hoc queries RD1 infers types from the values (whole numbers become integer, otherwise double). For DBI::dbReadTable() it reads the declared column types with PRAGMA table_info() and coerces columns to exactly what RSQLite would return, so whole tables round-trip faithfully.
- **No 64-bit integer tuning.** There is no bigint connection argument; integers beyond double precision may lose precision.
- **Booleans, dates, and times.** As in SQLite, these have no native type. Like RSQLite, logical maps to INTEGER and Date/POSIXct/difftime map to REAL on write, and are read back as integer/numeric rather than reconstructed.
- **Bound-parameter limit.** D1 caps bound parameters per request, so DBI::dbWriteTable() inserts rows in chunks.
- **Hidden internal tables.** DBI::dbListTables() omits D1/SQLite internal tables (sqlite_*, _cf_*).
- **Capabilities SQLite lacks.** Time travel and bookmarks (d1_bookmark(), d1_restore()), HTTP export/import (d1_export(), d1_import()), database management, and read replication.

Examples

```
## Not run:
con <- DBI::dbConnect(d1(), database_id = "....")
DBI::dbListTables(con)

## End(Not run)
```

---

d1_bookmark *Time travel: bookmarks and restoration*

Description

D1 retains write history so a database can be rewound to any point within its retention window. A *bookmark* is an opaque marker for a moment in that history. `d1_bookmark()` looks one up; `d1_restore()` rewinds the database.

Usage

```
d1_bookmark(
  database_id,
  timestamp = NULL,
  account_id = d1_account(),
  token = d1_token()
)

d1_restore(
  database_id,
  bookmark = NULL,
  timestamp = NULL,
  account_id = d1_account(),
  token = d1_token()
)
```

Arguments

`database_id`	A database UUID, name, `d1_database` object, or open `D1Connection` (resolved with `d1_database_id()`).
`timestamp`	An ISO 8601 timestamp (or `Date`/`POSIXct`). For `d1_bookmark()`, returns the bookmark at or just before this time (defaults to now). For `d1_restore()`, an alternative to `bookmark`.
`account_id, token`	Cloudflare credentials. See `d1_token()`.
`bookmark`	A bookmark string from `d1_bookmark()`.

Value

`d1_bookmark()` returns a bookmark string. `d1_restore()` returns a list with the new and previous bookmarks.

d1_database_id *Resolve a database name to its UUID*

Description

Passes a UUID through unchanged; otherwise looks up the database by exact name. Useful for calling any `d1_*()` function, or `DBI::dbConnect()`, with a human-readable name instead of a UUID.

Usage

```
d1_database_id(database, account_id = d1_account(), token = d1_token())
```

Arguments

`database`	A database UUID, name, a `d1_database` object from `d1_create_database()` / `d1_get_database()`, or an open `D1Connection`.
`account_id, token`	Cloudflare credentials. See `d1_token()`.

Value

A database UUID string.

d1_export *Export a D1 database*

Description

`d1_export()` triggers a SQL dump and polls until Cloudflare has staged the file, returning the result containing a signed download URL. `d1_download()` is a convenience wrapper that exports and downloads the SQL dump to a local file. `d1_download_sqlite()` goes one step further, materialising the dump into a local SQLite database that can be opened with RSQLite, mirroring the types of the live database.

Usage

```
d1_export(
  database_id,
  tables = NULL,
  no_data = FALSE,
  no_schema = FALSE,
  poll_interval = 1,
  account_id = d1_account(),
  token = d1_token()
```

```
)

d1_download(
  database_id,
  path,
  tables = NULL,
  account_id = d1_account(),
  token = d1_token()
)

d1_download_sqlite(
  database_id,
  path,
  tables = NULL,
  account_id = d1_account(),
  token = d1_token()
)
```

Arguments

database_id	A database UUID, name, d1_database object, or open D1Connection (resolved with d1_database_id()).
tables	Optional character vector restricting the export to these tables.
no_data	If TRUE, export only the schema.
no_schema	If TRUE, export only the data.
poll_interval	Seconds between polls while the export is prepared.
account_id, token	Cloudflare credentials. See d1_token().
path	Destination file path.

Value

d1_export() returns a list with filename and signed_url. d1_download() and d1_download_sqlite() return path invisibly.

---

d1_import *Import SQL into a D1 database*

Description

d1_import() uploads a local SQL file and ingests it, following Cloudflare's init/upload/ingest/poll flow. d1_upload_sqlite() is a convenience wrapper that dumps a local SQLite database to SQL and imports it, the inverse of d1_download_sqlite().

Usage

```
d1_import(
  database_id,
  file,
  poll_interval = 1,
  account_id = d1_account(),
  token = d1_token()
)

d1_upload_sqlite(
  database_id,
  file,
  account_id = d1_account(),
  token = d1_token()
)
```

Arguments

`database_id`	A database UUID, name, `d1_database` object, or open `D1Connection` (resolved with `d1_database_id()`).
`file`	Path to a `.sql` file (`d1_import()`) or `.sqlite` file (`d1_upload_sqlite()`).
`poll_interval`	Seconds between polls while the import is applied.
`account_id, token`	Cloudflare credentials. See `d1_token()`.

Value

A list describing the completed import (number of queries and final bookmark).

`d1_list_databases` *Manage D1 databases*

Description

Wrappers for the D1 database management endpoints.

Usage

```
d1_list_databases(name = NULL, account_id = d1_account(), token = d1_token())

d1_create_database(
  name,
  location_hint = NULL,
  account_id = d1_account(),
  token = d1_token()
)
```

```
d1_get_database(database_id, account_id = d1_account(), token = d1_token())

d1_delete_database(database_id, account_id = d1_account(), token = d1_token())

d1_set_replication(
  database_id,
  mode = c("auto", "disabled"),
  account_id = d1_account(),
  token = d1_token()
)
```

Arguments

name	Database name. For d1_list_databases(), an optional filter.
account_id, token	Cloudflare credentials. See d1_token().
location_hint	Optional primary location hint, e.g. "wnam".
database_id	A database UUID, name, d1_database object, or open D1Connection (resolved with d1_database_id()).
mode	Read-replication mode, "auto" or "disabled".

Value

d1_list_databases() returns a data frame; d1_create_database() and d1_get_database() return a database record (a list); d1_delete_database() returns NULL invisibly.

---

d1_pane *Show a D1 connection in the Connections pane*

Description

Registers an open connection with the RStudio / Positron Connections pane, listing its tables and adding bookmark, restore, and download actions. DBI::dbConnect() does this automatically; call d1_pane() to re-open the pane for an existing connection (for example after closing it).

Usage

```
d1_pane(conn)
```

Arguments

conn	A connection from DBI::dbConnect(d1(), ...).

Value

conn, invisibly.

d1_query *Run SQL against a D1 database*

Description

`d1_query()` calls the D1 `/query` endpoint and rectangles the response into a data frame. `d1_raw()` calls the `/raw` endpoint, which returns columns and rows as arrays; it is used internally by the DBI layer.

Usage

```
d1_query(
  database_id,
  sql,
  params = list(),
  account_id = d1_account(),
  token = d1_token()
)

d1_raw(
  database_id,
  sql,
  params = list(),
  account_id = d1_account(),
  token = d1_token()
)
```

Arguments

`database_id`	A database UUID, name, `d1_database` object, or open `D1Connection` (resolved with `d1_database_id()`).
`sql`	SQL string. May contain ? placeholders bound from `params`.
`params`	A list (or vector) of values bound to ? placeholders.
`account_id, token`	Cloudflare credentials. See `d1_token()`.

Details

A single SQL string may contain several statements separated by `;`. When it does, a list with one element per statement is returned; otherwise a single object is returned directly.

Value

For `d1_query()`, a data frame (or list of data frames). For `d1_raw()`, the parsed `result` list.

D1Connection-class *D1 DBI connection*

Description

A connection to a single D1 database. Created by `DBI::dbConnect()` with a `d1()` driver. Supports the usual DBI verbs: querying, listing and reading/writing tables. As D1 runs over stateless HTTP there are no transactions and disconnection is a no-op.

Slots

`database_id,account_id,token` Connection details.

`state` Environment tracking whether the connection is open.

D1Result-class *D1 DBI result*

Description

Represents the result of a statement run against a D1Connection. Because D1 returns the full result of a query in one HTTP response, the rows are materialised eagerly; `DBI::dbFetch()` then pages through them locally.

Slots

`conn` The originating connection.

`statement` The SQL statement.

`state` An environment holding the materialised rows and cursor.

Index