Package 'openlineage'

July 30, 2026

Title Create and Emit 'OpenLineage' Events

Version 0.1.0

Description Construct and validate run events that follow the 'OpenLineage' specification <https://openlineage.io/>. Model run lifecycles, datasets, and extensible facets with protocol-aware 'R' objects. Serialize events into deterministic JavaScript Object Notation (JSON) while preserving wire field names. Deliver events synchronously over Hypertext Transfer Protocol (HTTP) with authentication and retries, or use offline transports for local development and testing.

License MIT + file LICENSE

Depends R (>= 4.1.0)

Encoding UTF-8

Language en-US

RoxygenNote 8.0.0

Suggests knitr (>= 1.42), rmarkdown (>= 2.20), testthat (>= 3.2.2)

Config/testthat/edition 3

VignetteBuilder knitr

Imports cli (>= 3.0.0), httr2 (>= 1.0.0), jsonlite (>= 1.8.0), R6 (>= 2.5.0), S7 (>= 0.2.0), uuid (>= 1.1-0)

NeedsCompilation no

Author Pedro Baltazar [aut, cre, cph]

Maintainer Pedro Baltazar <pedrobtz@gmail.com>

Repository CRAN

Date/Publication 2026-07-30 16:30:02 UTC

Contents

`openlineage-package` *Create and Emit OpenLineage Events*

Description

Build validated OpenLineage 2.0.2 run events, serialize them to the protocol wire format, and emit them through local or synchronous HTTP transports.

Main workflow

Create a Run, Job, and optional input/output datasets, then combine them in a RunEvent. An OpenLineageClient emits the event through an injected transport or through HTTP configured by constructor arguments or environment variables.

Facets and extensions

Typed facet constructors cover common metadata. Use ol_facet to attach a facet schema that does not yet have a typed R constructor.

Author(s)

Maintainer: Pedro Baltazar <pedrobtz@gmail.com> [copyright holder]

Authors:

- Pedro Baltazar <pedrobtz@gmail.com> [copyright holder]

See Also

OpenLineageClient, RunEvent, local_transports, HttpTransport

`as_openlineage_list` *Convert an OpenLineage value to its wire representation*

Description

Recursively converts OpenLineage S7 models to named R lists. Protocol field names are applied here, `RunState` values are unwrapped, and `NULL` properties are omitted. Object key order is deterministic; array order is preserved.

Usage

```
as_openlineage_list(x)
```

Arguments

`x`	An OpenLineage model or supported nested value.

Value

A named list for a model, or the corresponding wire value for a nested value.

Examples

```
run <- Run("019c0000-0000-7000-8000-000000000001")
as_openlineage_list(run)
```

`datasets` *OpenLineage datasets*

Description

`Dataset` identifies a dataset. `InputDataset` and `OutputDataset` add lifecycle-specific facet maps and are the only dataset types accepted by a `RunEvent`.

Usage

```
Dataset(namespace, name, facets = list())

InputDataset(namespace, name, facets = list(), input_facets = list())

OutputDataset(namespace, name, facets = list(), output_facets = list())
```

Arguments

namespace	A non-empty namespace string.
name	A non-empty dataset name.
facets	A named list of dataset facets, or NULL to omit the field.
input_facets	A named list of input facets, or NULL to omit it.
output_facets	A named list of output facets, or NULL to omit it.

Value

A Dataset, InputDataset, or OutputDataset S7 object.

Examples

```
Dataset("postgres://warehouse", "analytics.orders")
InputDataset("postgres://warehouse", "raw.orders")
OutputDataset("postgres://warehouse", "analytics.orders")
```

---

dataset_facets *Typed dataset facets*

Description

Constructors for dataset-level schema, data-source, and tag metadata.

Usage

```
SchemaField(
  name,
  type = NULL,
  description = NULL,
  ordinal_position = NULL,
  fields = list()
)

SchemaDatasetFacet(
  fields = list(),
  producer = OPENLINEAGE_PRODUCER,
  deleted = NULL
)

DatasourceDatasetFacet(
  name = NULL,
  uri = NULL,
  producer = OPENLINEAGE_PRODUCER,
  deleted = NULL
)
```

Arguments

name	A field or data-source name.
type	An optional field type.
description	An optional field description.
ordinal_position	An optional one-based field position.
fields	For SchemaDatasetFacet, an unnamed list of SchemaField objects. For SchemaField, nested fields of the same type.
producer	A URI identifying the facet producer.
deleted	Whether this facet deletes a previously emitted facet.
uri	An optional data-source URI.

Value

A typed dataset facet or schema-field S7 object.

Examples

```
field <- SchemaField("order_id", "INTEGER", ordinal_position = 1L)
SchemaDatasetFacet(list(field))
DatasourceDatasetFacet("warehouse", "postgres://warehouse")
```

HttpTransport *Synchronous HTTP transport*

Description

Sends OpenLineage events as JSON POST requests. The base url and relative endpoint are joined with one slash; a URL that already ends in the endpoint is not duplicated.

Supply api_key for Authorization: Bearer <key> authentication, or place authentication headers in headers. Supplying both an API key and a custom Authorization header is an error. All custom headers are redacted from httr2 request printing.

HTTP 429, 500, 502, 503, and 504 responses and low-level connection failures are retried up to max_retries times. HTTP 401 and 403 responses are never retried. Set verify_tls = FALSE only for controlled development environments; doing so raises an openlineage_insecure_tls_warning.

Value

An R6 HTTP transport object.

Super class

OpenLineageTransport -> HttpTransport

Methods

Public methods:

- HttpTransport$new()
- HttpTransport$emit()
- HttpTransport$print()
- HttpTransport$clone()

`HttpTransport$new()`: Create an HTTP transport.

Usage:

```
HttpTransport$new(
  url,
  endpoint = "api/v1/lineage",
  api_key = NULL,
  headers = character(),
  timeout = 5,
  verify_tls = TRUE,
  max_retries = 3L
)
```

Arguments:

`url` An absolute HTTP or HTTPS base URL.

`endpoint` A relative OpenLineage endpoint path.

`api_key` An optional bearer API key.

`headers` Optional custom headers as a named character vector.

`timeout` A positive request timeout in seconds.

`verify_tls` Whether to verify TLS certificates.

`max_retries` A non-negative number of retries after the first request attempt.

Returns: A new `HttpTransport` object.

`HttpTransport$emit()`: Send an event to the configured OpenLineage HTTP endpoint.

Usage:

```
HttpTransport$emit(event)
```

Arguments:

`event` A `RunEvent` model.

Returns: `event`, invisibly, after a successful HTTP response.

`HttpTransport$print()`: Print a redacted transport summary.

Usage:

```
HttpTransport$print(...)
```

Arguments:

`...` Unused.

Returns: The transport, invisibly.

`HttpTransport$clone()`: The objects of this class are cloneable with this method.

Usage:

```
HttpTransport$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

local_transports

Examples

```
transport <- HttpTransport$new("https://example.com")
transport
```

`io_statistics_facets` *Input and output statistics facets*

Description

Input and output statistics facets

Usage

```
InputStatisticsInputDatasetFacet(
  row_count = NULL,
  size = NULL,
  file_count = NULL,
  producer = OPENLINEAGE_PRODUCER
)

OutputStatisticsOutputDatasetFacet(
  row_count = NULL,
  size = NULL,
  file_count = NULL,
  producer = OPENLINEAGE_PRODUCER
)
```

Arguments

`row_count`	The optional number of rows read or written.
`size`	The optional number of bytes read or written.
`file_count`	The optional number of files read or written.
`producer`	A URI identifying the facet producer.

Value

A typed input- or output-dataset facet S7 object.

Examples

```
InputStatisticsInputDatasetFacet(row_count = 100L)
OutputStatisticsOutputDatasetFacet(row_count = 100L, size = 2048L)
```

`Job` *OpenLineage job*

Description

Identifies a job within a producer-defined namespace.

Usage

```
Job(namespace, name, facets = list())
```

Arguments

`namespace`	A non-empty namespace string.
`name`	A non-empty job name.
`facets`	A named list of job facets, or `NULL` to omit the field.

Value

A `Job` S7 object.

Examples

```
Job("example-scheduler", "daily-report")
```

`job_facets` *Typed job facets*

Description

Constructors for SQL, source location, and job-type metadata.

Usage

```
SQLJobFacet(
  query,
  dialect = NULL,
  producer = OPENLINEAGE_PRODUCER,
  deleted = NULL
)

SourceCodeLocationJobFacet(
  type,
  url,
  repo_url = NULL,
  path = NULL,
  version = NULL,
  tag = NULL,
  branch = NULL,
  pull_request_number = NULL,
  producer = OPENLINEAGE_PRODUCER,
  deleted = NULL
)

EmissionPattern(event_trigger, event_content_mode, window_duration = NULL)

JobTypeJobFacet(
  processing_type,
  integration,
  job_type = NULL,
  emission_pattern = NULL,
  producer = OPENLINEAGE_PRODUCER,
  deleted = NULL
)
```

Arguments

query	The SQL query text.
dialect	An optional SQL dialect.
producer	A URI identifying the facet producer.
deleted	Whether this facet deletes a previously emitted facet.
type	The source-control system type.
url	The full source-code URL.
repo_url, path, version, tag, branch, pull_request_number	Optional source location details.
event_trigger	When events are emitted.
event_content_mode	Whether events accumulate or contain snapshots.

window_duration
: An optional positive duration in seconds.

processing_type
: The processing type, such as "BATCH".

integration
: The integration name, such as "DBT".

job_type
: An optional integration-specific job type.

emission_pattern
: An optional EmissionPattern.

Value

A typed job facet or emission-pattern S7 object.

Examples

```
SQLJobFacet("SELECT 1", dialect = "ansi")
SourceCodeLocationJobFacet(
  "git",
  "https://github.com/example/project/blob/main/job.R"
)
pattern <- EmissionPattern("EVENT_BASED", "COMPLETE_SNAPSHOT")
JobTypeJobFacet("BATCH", "R", emission_pattern = pattern)
```

---

local_transports *Local OpenLineage transports*

Description

Offline transports for testing, local development, and disabled emission. Every emit(event) method accepts a RunEvent model and returns that event invisibly after success.

Details

AccumulatingTransport stores emitted models in its public events list. Its clear() method removes all accumulated events and returns the transport invisibly.

ConsoleTransport writes one JSON event to stream. Set pretty = TRUE for indented output.

NoopTransport validates an event but performs no output or storage.

All transports provide close(), which returns TRUE invisibly because the local transports have no pending work.

Value

An R6 transport object.

Super class

OpenLineageTransport -> AccumulatingTransport

Public fields

`events` Emitted `RunEvent` models in emission order.

Methods

Public methods:

- `AccumulatingTransport$new()`
- `AccumulatingTransport$emit()`
- `AccumulatingTransport$clear()`
- `AccumulatingTransport$clone()`

`AccumulatingTransport$new()`: Create an empty accumulating transport.

Usage:

```
AccumulatingTransport$new()
```

Returns: A new `AccumulatingTransport` object.

`AccumulatingTransport$emit()`: Store an event.

Usage:

```
AccumulatingTransport$emit(event)
```

Arguments:

`event` A `RunEvent` model.

Returns: `event`, invisibly.

`AccumulatingTransport$clear()`: Remove all stored events.

Usage:

```
AccumulatingTransport$clear()
```

Returns: The transport, invisibly.

`AccumulatingTransport$clone()`: The objects of this class are cloneable with this method.

Usage:

```
AccumulatingTransport$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Super class

`OpenLineageTransport` -> `ConsoleTransport`

Methods

Public methods:

- ConsoleTransport$new()
- ConsoleTransport$emit()
- ConsoleTransport$clone()

`ConsoleTransport$new()`: Create a console transport.

Usage:

```
ConsoleTransport$new(stream = stdout(), pretty = TRUE)
```

Arguments:

`stream` An open, writable R connection.

`pretty` Whether JSON should be indented.

Returns: A new `ConsoleTransport` object.

`ConsoleTransport$emit()`: Serialize and write an event to the configured connection.

Usage:

```
ConsoleTransport$emit(event)
```

Arguments:

`event` A `RunEvent` model.

Returns: `event`, invisibly.

`ConsoleTransport$clone()`: The objects of this class are cloneable with this method.

Usage:

```
ConsoleTransport$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Super class

`OpenLineageTransport` -> `NoopTransport`

Methods

Public methods:

- NoopTransport$emit()
- NoopTransport$clone()

`NoopTransport$emit()`: Validate an event without emitting it.

Usage:

```
NoopTransport$emit(event)
```

Arguments:

`event` A `RunEvent` model.

Returns: `event`, invisibly.

`NoopTransport$clone()`: The objects of this class are cloneable with this method.

Usage:

```
NoopTransport$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
transport <- AccumulatingTransport$new()
event <- RunEvent(
  Run(new_run_id()),
  Job("example", "task"),
  event_type = "START"
)
transport$emit(event)
length(transport$events)

console <- ConsoleTransport$new(pretty = FALSE)
noop <- NoopTransport$new()
noop$emit(event)
```

`new_event_time` *Format an OpenLineage event time*

Description

Converts an R date-time to an ISO 8601 UTC timestamp with millisecond precision, suitable for the `eventTime` field of an OpenLineage event.

Usage

```
new_event_time(time = Sys.time())
```

Arguments

`time`	A single, non-missing `POSIXct` or `POSIXlt` value. Defaults to the current time.

Value

A single UTC timestamp string.

Examples

```
time <- as.POSIXct("2026-01-02 03:04:05", tz = "UTC")
new_event_time(time)
```

new_run_id *Generate a run identifier*

Description

Generates a random UUID suitable for the `runId` field of an OpenLineage run. The UUID version is an implementation detail and may change.

Usage

```
new_run_id()
```

Value

A single UUID string.

Examples

```
new_run_id()
```

ol_facet *Create a generic OpenLineage facet*

Description

Creates a facet for extensions or facet schemas without a typed constructor. Names supplied through `...` are preserved exactly at the JSON boundary.

Usage

```
ol_facet(schema_url, ..., producer = OPENLINEAGE_PRODUCER, deleted = NULL)
```

Arguments

`schema_url`	The URI of the facet's JSON Schema definition.
`...`	Named facet fields using their OpenLineage wire names.
`producer`	A URI identifying the facet producer.
`deleted`	Whether this facet deletes a previously emitted job or dataset facet. Use only where the target schema supports `_deleted`.

Value

A generic OpenLineage facet accepted in any facet map.

Examples

```
facet <- ol_facet(
  "https://example.com/CustomRunFacet.json",
  customValue = "example"
)
Run(new_run_id(), facets = list(custom = facet))
```

---

ol_tag *Create an OpenLineage tag*

Description

Create an OpenLineage tag

Usage

```
ol_tag(key, value, source = NULL, field = NULL)
```

Arguments

key	The tag key.
value	The tag value.
source	An optional tag source.
field	An optional dataset field to which the tag applies.

Value

A tag record for a typed tags facet.

Examples

```
ol_tag("environment", "production", source = "USER")
```

OpenLineageClient *OpenLineage client*

Description

Coordinates event emission through an OpenLineage transport. Supply a transport directly for local or custom delivery, or configure the built-in HTTP transport with `url` and related arguments.

When an argument is absent, the client reads `OPENLINEAGE_URL`, `OPENLINEAGE_ENDPOINT`, `OPENLINEAGE_API_KEY`, and `OPENLINEAGE_DISABLED`. Explicit arguments take precedence over their environment equivalents. `OPENLINEAGE_DISABLED` accepts only `true` or `false`, ignoring case and surrounding whitespace.

A disabled client uses NoopTransport. An injected `transport` takes precedence over HTTP environment variables. If neither a transport nor URL is configured, the client uses ConsoleTransport. Supplying partial HTTP settings without a URL is an error.

Value

An R6 OpenLineage client object.

Public fields

`transport` The configured OpenLineage transport.

Methods

Public methods:

- `OpenLineageClient$new()`
- `OpenLineageClient$emit()`
- `OpenLineageClient$print()`
- `OpenLineageClient$clone()`

`OpenLineageClient$new()`: Create an OpenLineage client.

Usage:

```
OpenLineageClient$new(
  transport = NULL,
  url = NULL,
  endpoint = "api/v1/lineage",
  api_key = NULL,
  headers = character(),
  timeout = 5,
  verify_tls = TRUE,
  max_retries = 3L,
  disabled = NULL
)
```

Arguments:

`transport` An optional R6 transport with an `emit(event)` method.

`url` An optional HTTP or HTTPS base URL. If `NULL`, `OPENLINEAGE_URL` is used when set.

`endpoint` A relative OpenLineage endpoint path. If omitted, `OPENLINEAGE_ENDPOINT` is used when set.

`api_key` An optional bearer API key. If `NULL`, `OPENLINEAGE_API_KEY` is used when set.

`headers` Optional custom HTTP headers as a named character vector.

`timeout` A positive HTTP request timeout in seconds.

`verify_tls` Whether to verify TLS certificates.

`max_retries` A non-negative number of HTTP retries after the first request attempt.

`disabled` Whether to disable emission. If `NULL`, `OPENLINEAGE_DISABLED` is used when set.

Returns: A new `OpenLineageClient` object.

`OpenLineageClient$emit()`: Emit an OpenLineage event through the configured transport.

Usage:

```
OpenLineageClient$emit(event)
```

Arguments:

`event` A `RunEvent` model.

Returns: `event`, invisibly, after successful delivery.

`OpenLineageClient$print()`: Print a credential-free client summary.

Usage:

```
OpenLineageClient$print(...)
```

Arguments:

`...` Unused.

Returns: The client, invisibly.

`OpenLineageClient$clone()`: The objects of this class are cloneable with this method.

Usage:

```
OpenLineageClient$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

See Also

local_transports, HttpTransport

Examples

```
transport <- AccumulatingTransport$new()
client <- OpenLineageClient$new(transport = transport, disabled = FALSE)
event <- RunEvent(
  Run(new_run_id()),
  Job("example", "task"),
  event_type = "START"
)
client$emit(event)
length(transport$events)
```

`openlineage_constants` *OpenLineage protocol constants*

Description

Constants used by `openlineage` when constructing OpenLineage events. `OPENLINEAGE_SCHEMA_URL` identifies the schema document, while `OPENLINEAGE_RUN_EVENT_SCHEMA_URL` identifies its `RunEvent` definition.

Usage

```
OPENLINEAGE_SCHEMA_VERSION

OPENLINEAGE_SCHEMA_URL

OPENLINEAGE_RUN_EVENT_SCHEMA_URL

OPENLINEAGE_PRODUCER
```

Format

Character scalars.

Examples

```
OPENLINEAGE_SCHEMA_VERSION
OPENLINEAGE_SCHEMA_URL
OPENLINEAGE_RUN_EVENT_SCHEMA_URL
OPENLINEAGE_PRODUCER
```

Run *OpenLineage run*

Description

Identifies one execution of an OpenLineage job.

Usage

```
Run(run_id, facets = list())
```

Arguments

`run_id`	A valid UUID string.
`facets`	A named list of run facets, or `NULL` to omit the field.

Value

A `Run` S7 object.

Examples

```
Run(new_run_id())
```

RunEvent *OpenLineage run event*

Description

Describes a lifecycle transition for one job run and its input and output datasets.

Usage

```
RunEvent(
  run,
  job,
  event_type = NULL,
  event_time = new_event_time(),
  inputs = list(),
  outputs = list(),
  producer = OPENLINEAGE_PRODUCER,
  schema_url = OPENLINEAGE_RUN_EVENT_SCHEMA_URL
)
```

Arguments

run	A Run object.
job	A Job object.
event_type	A RunState, a valid state string, or NULL to omit it.
event_time	An RFC 3339 date-time string with a time-zone offset.
inputs	An unnamed list of InputDataset objects, or NULL.
outputs	An unnamed list of OutputDataset objects, or NULL.
producer	A URI identifying the event producer.
schema_url	The RunEvent JSON Schema URL.

Value

A RunEvent S7 object.

Examples

```
run <- Run(new_run_id())
job <- Job("example-scheduler", "daily-report")
RunEvent(
  run,
  job,
  event_type = "START",
  event_time = new_event_time(as.POSIXct("2026-01-02", tz = "UTC"))
)
```

RunState *OpenLineage run state*

Description

A validated lifecycle state for an OpenLineage run.

Usage

```
RunState(value)
```

Arguments

value	One of "START", "RUNNING", "COMPLETE", "ABORT", "FAIL", or "OTHER".

Value

A RunState S7 object.

Examples

```
RunState("START")
```

run_facets *Typed run facets*

Description

Constructors for commonly used OpenLineage run facets.

Usage

```
NominalTimeRunFacet(
  nominal_start_time,
  nominal_end_time = NULL,
  producer = OPENLINEAGE_PRODUCER
)

ParentRunFacet(run, job, root = NULL, producer = OPENLINEAGE_PRODUCER)

ErrorMessageRunFacet(
  message,
  programming_language,
  stack_trace = NULL,
  producer = OPENLINEAGE_PRODUCER
)
```

```
ProcessingEngineRunFacet(
  version,
  name = NULL,
  openlineage_adapter_version = NULL,
  producer = OPENLINEAGE_PRODUCER
)
```

Arguments

nominal_start_time	The nominal start as an RFC 3339 date-time.
nominal_end_time	An optional nominal end date-time.
producer	A URI identifying the facet producer.
run	The parent Run.
job	The parent Job.
root	Optional list(run = Run(...), job = Job(...)) for the root.
message	A human-readable error message.
programming_language	The language that produced the error.
stack_trace	An optional stack trace.
version	The processing engine version.
name	The optional processing engine name.
openlineage_adapter_version	The optional adapter version.

Value

A typed run facet S7 object.

Examples

```
NominalTimeRunFacet("2026-01-02T03:00:00.000Z")
parent_run <- Run(new_run_id())
parent_job <- Job("example", "parent")
ParentRunFacet(parent_run, parent_job)
ErrorMessageRunFacet("Query failed", "R")
ProcessingEngineRunFacet("4.6.0", name = "R")
```

tags_facets *Typed tags facets*

Description

Typed tags facets

Usage

```
TagsJobFacet(tags = list(), producer = OPENLINEAGE_PRODUCER, deleted = NULL)

TagsRunFacet(tags = list(), producer = OPENLINEAGE_PRODUCER)

TagsDatasetFacet(
  tags = list(),
  producer = OPENLINEAGE_PRODUCER,
  deleted = NULL
)
```

Arguments

tags	An unnamed list created with ol_tag().
producer	A URI identifying the facet producer.
deleted	Whether a job or dataset facet is being deleted.

Value

A typed tags facet S7 object.

Examples

```
tag <- ol_tag("environment", "production")
TagsJobFacet(list(tag))
TagsRunFacet(list(tag))
TagsDatasetFacet(list(
  ol_tag("sensitivity", "restricted", field = "customer_id")
))
```

`to_openlineage_json` *Serialize an OpenLineage value to JSON*

Description

Serialize an OpenLineage value to JSON

Usage

```
to_openlineage_json(x, pretty = FALSE)
```

Arguments

`x`	An OpenLineage model or supported nested value.
`pretty`	Whether to add indentation and line breaks.

Value

A single JSON string.

Examples

```
event <- RunEvent(
  Run("019c0000-0000-7000-8000-000000000001"),
  Job("example", "task"),
  event_type = "START",
  event_time = "2026-01-02T03:04:05.000Z"
)
to_openlineage_json(event)
```

Index