

# Package ‘gtfsio’

October 20, 2023

**Type** Package

**Title** Read and Write General Transit Feed Specification (GTFS) Files

**Version** 1.1.1

**Description** Tools for the development of packages related to General Transit Feed Specification (GTFS) files. Establishes a standard for representing GTFS feeds using R data types. Provides fast and flexible functions to read and write GTFS feeds while sticking to this standard. Defines a basic 'gtfs' class which is meant to be extended by packages that depend on it. And offers utility functions that support checking the structure of GTFS objects.

**License** MIT + file LICENSE

**URL** <https://r-transit.github.io/gtfsio/>,  
<https://github.com/r-transit/gtfsio>

**BugReports** <https://github.com/r-transit/gtfsio/issues>

**Imports** data.table, utils, zip

**Suggests** knitr, rmarkdown, tinytest

**VignetteBuilder** knitr

**Encoding** UTF-8

**RoxygenNote** 7.2.3

**Collate** 'gtfsio\_error.R' 'assert\_gtfs.R' 'assert\_inputs.R' 'checks.R'  
'export\_gtfs.R' 'get\_gtfs\_standards.R' 'gtfs\_methods.R'  
'gtfs\_subset.R' 'gtfsio.R' 'import\_gtfs.R' 'new\_gtfs.R'

**NeedsCompilation** no

**Author** Daniel Herszenhut [aut, cre] (<<https://orcid.org/0000-0001-8066-1105>>),  
Flavio Poletti [aut],  
Mark Padgham [aut],  
Rafael H. M. Pereira [rev] (<<https://orcid.org/0000-0003-2125-7465>>),  
Tom Buckley [rev],  
Ipea - Institute for Applied Economic Research [cph, fnd]

**Maintainer** Daniel Herszenhut <dhersz@gmail.com>

**Repository** CRAN

**Date/Publication** 2023-10-20 04:50:23 UTC

## R topics documented:

assert_gtfs . . . . .	2
check_field_class . . . . .	3
check_field_exists . . . . .	4
check_file_exists . . . . .	5
export_gtfs . . . . .	5
get_gtfs_standards . . . . .	7
import_gtfs . . . . .	8
new_gtfs . . . . .	9
print.gtfs . . . . .	11
summary.gtfs . . . . .	12
[.gtfs . . . . .	12
<b>Index</b>	<b>14</b>

---

assert_gtfs	<i>GTFS object validator</i>
-------------	------------------------------

---

### Description

Asserts that a GTFS object is valid. Valid objects are those in which:

- Every element is named.
- Every element inherits from `data.frames`.

The exception to the second rule are objects that contain an element named `"."`. In such case, this element is actually composed by a named list of elements who inherit from `data.frames`.

### Usage

```
assert_gtfs(x)
```

### Arguments

`x` A GTFS object.

### Value

The same GTFS object passed to `x`.

### See Also

Other constructors: [new\\_gtfs\(\)](#)

## Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

gtfs <- import_gtfs(gtfs_path)
gtfs <- assert_gtfs(gtfs)
```

---

check_field_class	<i>Check the classes of fields in a GTFS object element</i>
-------------------	---

---

## Description

Checks the classes of fields, represented by columns, inside a GTFS object element.

## Usage

```
check_field_class(x, file, fields, classes)

assert_field_class(x, file, fields, classes)
```

## Arguments

x	A GTFS object.
file	A string. The element, that represents a GTFS text file, whose fields' classes should be checked.
fields	A character vector. The fields to have their classes checked.
classes	A character vector, with the same length of fields. The classes that each field must inherit from.

## Value

check\_field\_class returns TRUE if the check is successful, and FALSE otherwise.  
assert\_field\_class returns x invisibly if the check is successful, and throws an error otherwise.

## See Also

Other checking functions: [check\\_field\\_exists\(\)](#), [check\\_file\\_exists\(\)](#)

## Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

check_field_class(
  gtfs,
  "calendar",
  fields = c("monday", "tuesday"),
```

```

  classes = rep("integer", 2)
)

check_field_class(
  gtfs,
  "calendar",
  fields = c("monday", "tuesday"),
  classes = c("integer", "character")
)

```

---

check\_field\_exists      *Check the existence of fields in a GTFS object element*

---

### Description

Checks the existence of fields, represented by columns, inside a GTFS object element.

### Usage

```

check_field_exists(x, file, fields)

assert_field_exists(x, file, fields)

```

### Arguments

x	A GTFS object.
file	A string. The element, that represents a GTFS text file, where fields should be searched.
fields	A character vector. The fields to check the existence of.

### Value

check\_field\_exists returns TRUE if the check is successful, and FALSE otherwise.  
 assert\_field\_exists returns x invisibly if the check is successful, and throws an error otherwise.

### See Also

Other checking functions: [check\\_field\\_class\(\)](#), [check\\_file\\_exists\(\)](#)

### Examples

```

gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

check_field_exists(gtfs, "calendar", c("monday", "tuesday"))

check_field_exists(gtfs, "calendar", c("monday", "oi"))

```

---

check_file_exists	<i>Check the existence of text files in a GTFS object</i>
-------------------	---

---

**Description**

Checks the existence of elements inside a GTFS object that represent specific GTFS text files.

**Usage**

```
check_file_exists(x, files)
```

```
assert_file_exists(x, files)
```

**Arguments**

x	A GTFS object.
files	A character vector. The files to check the existence of.

**Value**

check\_file\_exists returns TRUE if the check is successful, and FALSE otherwise.  
assert\_file\_exists returns x invisibly if the check is successful, and throws an error otherwise.

**See Also**

Other checking functions: [check\\_field\\_class\(\)](#), [check\\_field\\_exists\(\)](#)

**Examples**

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

check_file_exists(gtfs, c("calendar", "agency"))

check_file_exists(gtfs, c("calendar", "oi"))
```

---

export_gtfs	<i>Export GTFS objects</i>
-------------	----------------------------

---

**Description**

Writes GTFS objects to disk as GTFS transit feeds. The object must be formatted according to the standards for reading and writing GTFS transit feeds, as specified in [get\\_gtfs\\_standards](#) (i.e. data types are not checked). If present, does not write auxiliary tables held in a sub-list named ". ".

**Usage**

```
export_gtfs(
  gtfs,
  path,
  files = NULL,
  standard_only = FALSE,
  compression_level = 9,
  as_dir = FALSE,
  overwrite = TRUE,
  quiet = TRUE
)
```

**Arguments**

<code>gtfs</code>	A GTFS object.
<code>path</code>	A string. Where the resulting .zip file must be written to.
<code>files</code>	A character vector. The name of the elements to be written to the feed.
<code>standard_only</code>	A logical. Whether only standard files and fields should be written (defaults to TRUE, which drops extra files and fields).
<code>compression_level</code>	A numeric, between 1 and 9. The higher the value, the best the compression, which demands more processing time. Defaults to 9 (best compression).
<code>as_dir</code>	A logical. Whether the feed should be exported as a directory, instead of a .zip file. Defaults to FALSE.
<code>overwrite</code>	A logical. Whether to overwrite an existing .zip file (defaults to TRUE).
<code>quiet</code>	A logical. Whether to hide log messages and progress bars (defaults to TRUE).

**Value**

Invisibly returns the same GTFS object passed to `gtfs`.

**See Also**

[get\\_gtfs\\_standards](#)

Other io functions: [import\\_gtfs\(\)](#)

**Examples**

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

gtfs <- import_gtfs(gtfs_path)

tmpf <- tempfile(pattern = "gtfs", fileext = ".zip")

export_gtfs(gtfs, tmpf)
zip::zip_list(tmpf)$filename
```

```
export_gtfs(gtfs, tmpf, files = c("shapes", "trips"))
zip::zip_list(tmpf)$filename
```

---

get\_gtfs\_standards      *Generate GTFS standards*

---

## Description

Generates a list specifying the standards to be used when reading and writing GTFS feeds with R. Each list element (also a list) represents a distinct GTFS table, and describes:

- whether the table is required, optional or conditionally required;
- the fields that compose the table, including which R data type is best suited to represent it, whether the field is required, optional or conditionally required, and which values it can assume (most relevant to GTFS ENUMs).

Note: the standards list is based on the specification as revised in May 9th, 2022.

## Usage

```
get_gtfs_standards()
```

## Value

A named list, in which each element represents the R equivalent of each GTFS table standard.

## Details

GTFS standards were derived from [GTFS Schedule Reference](#). The R data types chosen to represent each GTFS data type are described below:

- Color = character
- Currency amount = numeric
- Currency code = character
- Date = integer
- Email = character
- ENUM = integer
- ID = character
- Integer = integer
- Language code = character
- Latitude = numeric
- Longitude = numeric
- Float = numeric
- Phone number = character

- Text = character
- Time = character
- Timezone = character
- URL = character

### Examples

```
gtfs_standards <- get_gtfs_standards()
```

---

import\_gtfs

*Import GTFS transit feeds*

---

### Description

Imports GTFS transit feeds from either a local .zip file or an URL. Columns are parsed according to the standards for reading and writing GTFS feeds specified in [get\\_gtfs\\_standards](#).

### Usage

```
import_gtfs(
  path,
  files = NULL,
  fields = NULL,
  extra_spec = NULL,
  skip = NULL,
  quiet = TRUE,
  encoding = "unknown"
)
```

### Arguments

path	A string. The path to a GTFS .zip file.
files	A character vector. The text files to be read from the GTFS, without the .txt extension. If NULL (the default), all existing text files are read.
fields	A named list. The fields to be read from each text file, in the format <code>list(file1 = c("field1", "field2"))</code> . If NULL (the default), all fields from the files specified in files are read. If a file is specified in files but not in fields, all fields from that file will be read (i.e. you may specify in fields only files whose fields you want to subset).
extra_spec	A named list. Custom specification used when reading undocumented fields, in the format <code>list(file1 = c(field1 = "type1", field2 = "type2"))</code> . If NULL (the default), all undocumented fields are read as character. Similarly, if an undocumented field is not specified in extra_spec, it is read as character (i.e. you may specify in extra_spec only the fields that you want to read as a different type). Only supports the character, integer and numeric types, also used in <a href="#">get_gtfs_standards</a> .



skip	A character vector. Text files that should not be read from the GTFS, without the .txt extension. If NULL (the default), no files are skipped. Cannot be used if files is set.
quiet	A logical. Whether to hide log messages and progress bars (defaults to TRUE).
encoding	A string. Passed to <code>fread</code> , defaults to "unknown". Other possible options are "UTF-8" and "Latin-1". Please note that this is not used to re-encode the input, but to enable handling encoded strings in their native encoding.

### Value

A GTFS object: a named list of data frames, each one corresponding to a distinct text file from the given GTFS feed.

### See Also

[get\\_gtfs\\_standards](#)

Other io functions: [export\\_gtfs\(\)](#)

### Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

# read all files and columns
gtfs <- import_gtfs(gtfs_path)
names(gtfs)
names(gtfs$trips)

# read all columns from selected files
gtfs <- import_gtfs(gtfs_path, files = c("trips", "stops"))
names(gtfs)
names(gtfs$trips)

# read specific columns from selected files
gtfs <- import_gtfs(
  gtfs_path,
  files = c("trips", "stops"),
  fields = list(
    trips = c("route_id", "trip_id"),
    stops = c("stop_id", "stop_lat", "stop_lon")
  )
)
```

**Description**

Creates a GTFS object. Mostly useful for package authors who may want to either create `gtfs` objects in their packages or create subclasses of the main `gtfs` class. The usage of this function assumes some knowledge on `gtfs` objects, thus inputs are not extensively checked. See [assert\\_gtfs](#) for more thorough checks.

**Usage**

```
new_gtfs(x, subclass = character(), ...)
```

**Arguments**

<code>x</code>	A GTFS-like object (either a GTFS object or a named list). Each element must represent a distinct GTFS text file.
<code>subclass</code>	A character vector. Subclasses to be assigned to the <code>gtfs</code> object.
<code>...</code>	Name-value pairs. Additional attributes.

**Value**

A GTFS object: a named list of data frames, each one corresponding to a distinct GTFS text file, with `gtfs` and `list` classes.

**See Also**

Other constructors: [assert\\_gtfs\(\)](#)

**Examples**

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")

tmpdir <- tempfile(pattern = "new_gtfs_example")
zip::unzip(gtfs_path, exdir = tmpdir)

agency <- data.table::fread(file.path(tmpdir, "agency.txt"))
stops <- data.table::fread(file.path(tmpdir, "stops.txt"))
routes <- data.table::fread(file.path(tmpdir, "routes.txt"))
trips <- data.table::fread(file.path(tmpdir, "trips.txt"))
stop_times <- data.table::fread(file.path(tmpdir, "stop_times.txt"))
calendar <- data.table::fread(file.path(tmpdir, "calendar.txt"))

txt_files <- list(
  agency = agency,
  stops = stops,
  routes = routes,
  trips = trips,
  stop_times = stop_times,
  calendar = calendar
)

gtfs <- new_gtfs(txt_files)
```

```
class(gtfs)
names(gtfs)
```

---

print.gtfs	<i>Print a GTFS object</i>
------------	----------------------------

---

### Description

Prints a GTFS object suppressing the class attribute.

### Usage

```
## S3 method for class 'gtfs'
print(x, ...)
```

### Arguments

x	A GTFS object.
...	Optional arguments ultimately passed to format.

### Value

The GTFS object that was printed, invisibly.

### Examples

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

# subset 'gtfs' for a smaller output
gtfs <- gtfs[c("routes", "trips")]

print(gtfs)
```

---

summary.gtfs	<i>Print summary of a GTFS object</i>
--------------	---------------------------------------

---

**Description**

Print summary of a GTFS object

**Usage**

```
## S3 method for class 'gtfs'
summary(object, ...)
```

**Arguments**

object	A GTFS object.
...	Ignored.

**Examples**

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
gtfs <- import_gtfs(gtfs_path)

summary(gtfs)
```

---

[.gtfs	<i>Subset a GTFS object</i>
--------	-----------------------------

---

**Description**

Subsetting a GTFS object using [ preserves the object class.

**Usage**

```
## S3 method for class 'gtfs'
x[value]
```

**Arguments**

x	A GTFS object.
value	Either a numeric or a character vector. Designates the elements to be returned.

**Value**

A GTFS object.

**Examples**

```
gtfs_path <- system.file("extdata/ggl_gtfs.zip", package = "gtfsio")
```

```
gtfs <- import_gtfs(gtfs_path)
names(gtfs)
class(gtfs)
```

```
small_gtfs <- gtfs[1:5]
names(small_gtfs)
class(small_gtfs)
```

```
small_gtfs <- gtfs[c("shapes", "trips")]
names(small_gtfs)
class(small_gtfs)
```

# Index

## \* **checking functions**

- check\_field\_class, 3
- check\_field\_exists, 4
- check\_file\_exists, 5

## \* **constructors**

- assert\_gtfs, 2
- new\_gtfs, 9

## \* **io functions**

- export\_gtfs, 5
- import\_gtfs, 8

[.gtfs, 12

assert\_field\_class (check\_field\_class),  
3

assert\_field\_exists  
(check\_field\_exists), 4

assert\_file\_exists (check\_file\_exists),  
5

assert\_gtfs, 2, 10

check\_field\_class, 3, 4, 5

check\_field\_exists, 3, 4, 5

check\_file\_exists, 3, 4, 5

export\_gtfs, 5, 9

fread, 9

get\_gtfs\_standards, 5, 6, 7, 8, 9

import\_gtfs, 6, 8

new\_gtfs, 2, 9

print.gtfs, 11

summary.gtfs, 12