# Package: MsStash (via r-universe)

May 12, 2026

**Title** Infrastructure to serialize and restore mass spectrometry data objects

**Version** 0.97.0

**Description** The serialization mechanism of R allows to save and load R data objects in a binary format, that can however not be read by other programming languages or software. The MsStash package defines classes and method to enable serializing and restoring, or importing, mass spectrometry data objects to and from language agnostic file formats. A variety of different file types, including HDF5 and JSON-based formats defined by the Bioconductor *alabaster* package are supported. The file type can be defined and configured through a second argument `param` of the save/read methods.

**Depends** R (>= 4.6.0)

**Imports** methods, ProtGenerics

**Suggests** BiocStyle, rmarkdown, knitr, alabaster.base, fs, testthat

**License** Artistic-2.0

**Encoding** UTF-8

**VignetteBuilder** knitr

**BugReports** https://github.com/RforMassSpectrometry/MsStash/issues

**URL** https://github.com/RforMassSpectrometry/MsStash

**biocViews** Infrastructure, MassSpectrometry, Metabolomics, DataImport, Proteomics

**Roxygen** list(markdown=TRUE)

**RoxygenNote** 7.3.3

**Collate** 'PlainTextParam.R' 'AlabasterParam.R' 'AllGenerics.R'

**Repository** https://rformassspectrometry.r-universe.dev

**Date/Publication** 2026-05-12 09:16:11 UTC

**RemoteUrl** https://github.com/rformassspectrometry/MsStash

**RemoteRef** HEAD

**RemoteSha** 633c0cb656271924e11f59ca708326cbed04962d

# Contents



---

`AlabasterParam` *Store MS data objects using the alabaster framework*

---

## Description

The *alabaster* framework provides the methodology to save R objects to on-disk representations/ storage modes which are programming language independent (in contrast to e.g. R's RDS files). By using standard file formats such as JSON and HDF5, alabaster ensures that the data can also be read and imported by other programming languages such as Python or Javascript. This improves interoperability between application ecosystems.

The *alabaster* package defines the `alabaster.base::saveObject()` and `alabaster.base::readObject()` methods that have to be implemented for specific data classes to enable saving to or reading from alabaster-based file formats.

In addition, the *MsStash* package defines the `AlabasterParam` which can be used to write or read MS objects using the `saveMsObject()` and `readMsObject()` methods in alabaster format. This allows additional configurations and customizations to the export or import process. It is thus for example possible to specify the path to the original MS data files for *on-disk* MS representations such as the `Spectra::MsBackendMzR` which enables to import a stored object even if either the object or the original MS data files have been moved to a different directory or file system.

In order to enable alabaster-based import/export, the respective methods have to be implemented for the class. See the package vignette for examples and details.

## Usage

```
AlabasterParam(path = tempdir())
```

## Arguments

| | |
|---|---|
| `path` | `character(1)` with the name of the directory where the MS data object should be saved to or from which it should be restored. Importantly, path should point to a **new** folder, i.e. a directory that **does not already exist**. |

## Details

Importantly, it is only possible to save **one object in one directory**. To overwrite an existing stored object in a folder, that folder has to be deleted beforehand.

## Value

For `AlabasterParam()`: an instance of `AlabasterParam` class. For `readObject()` the exported object in the specified path (depending on the type of object defined in the *OBJECT* file in the path. For `readMsObject()` the exported data object, defined with the function's first parameter, from the specified path. `saveObject()` and `saveMsObject()` don't return anything.

## Author(s)

Johannes Rainer, Philippine Louail

## See Also

Other MS object export and import formats.: `PlainTextParam`

## Examples

```
## Create an AlabasterParam object
a <- AlabasterParam(path = tempdir())
a

## See the package vignette for example implemetations and usage.
```

---

# PlainTextParam *Store contents of MS objects as plain text files*

---

## Description

The `saveMsObject()` and `readMsObject()` methods with the `PlainTextParam` option enable users to save/load different type of mass spectrometry (MS) object as a collections of plain text files in/from a specified folder. This folder, defined with the `path` parameter, will be created by the `saveMsObject()` function. Writing data to a folder that contains already exported data will result in an error.

For `PlainTextParam` all data is expected to be exported to plain text files, where possible as tabulator delimited text files.

To support writing/reading with `PlainTextParam`, the `saveMsData()` and `readMsData()` methods have to be implemented for the respective class.

See the package vignette for example implementations and details.

## Usage

```
PlainTextParam(path = tempdir())
```

## Arguments

| | |
|---|---|
| `path` | For `PlainTextParam()`: `character(1)`, defining where the files are going to be stored/ should be loaded from. The default is `path = tempdir()`. |

## Value

For `PlainTextParam()`: a `PlainTextParam` class. `saveMsObject()` does not return anything but saves the object to collections of different plain text files to a folder. The `readMsObject()` method returns the restored data as an instance of the class specified with parameter `object`.

## Author(s)

Philippine Louail, Johannes Rainer

## See Also

Other MS object export and import formats.: `AlabasterParam`

## Examples

```
## Create a PlainTextParam object
p <- PlainTextParam()
p

## For example implementations and details see the package vignette
```

---

`saveMsObject` *Save and load MS data objects to and from different file formats*

---

## Description

The `saveMsObject()` and `readMsObject()` methods allow serializing and restoring/importing mass spectrometry (MS) data (or result) objects to and from language agnostic file formats. The type and configuration of the file format is defined by the second argument to the method, `param`.

- `saveMsObject(object, param)`: saves the MS data object `object` to file(s) in a format defined by `param`.
- `readMsObject(object, param)`: `object` defines the type of MS object that should be returned by the function and `param` the format and file name(s) from which the data should be restored/imported.

Examples implementations of these methods are presented in the package vignette.

## Usage

```
saveMsObject(object, param, ...)

readMsObject(object, param, ...)
```

## Arguments

| | |
|---|---|
| `object` | for `saveMsObject()`: the MS data object to save, for `readMsObject()`: the MS data object that should be returned |
| `param` | an object defining and (eventually configuring) the file format and file name or directory to/from which the data object should be exported/imported. |
| `...` | additional optional arguments. See documentation of respective method for more information. |

## Value

`saveMsObject()` has no return value, `readMsObject` is expected to return an instance of the class defined with `object`.

## Author(s)

Philippine Louail, Johannes Rainer, Laurent Gatto

# Index