

# Package: vscDebugger (via r-universe)

January 10, 2025

**Title** Support for Visual Studio Code Debugger

**Version** 0.5.3

**Description** Provides support for a visual studio code debugger

**License** MIT

**URL** <https://ManuelHentschel.github.io/vscDebugger/>,  
<https://github.com/ManuelHentschel/vscDebugger>

**BugReports** <https://github.com/ManuelHentschel/vscDebugger/issues>

**Imports** jsonlite, R6, tcltk

**Suggests** pkgload, knitr, rmarkdown, markdown

**Encoding** UTF-8

**LazyData** true

**NeedsCompilation** yes

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.1

**VignetteBuilder** knitr

**Repository** <https://opensafely-core.r-universe.dev>

**RemoteUrl** <https://github.com/ManuelHentschel/vscDebugger>

**RemoteRef** HEAD

**RemoteSha** 83e91154629de8cb81d043dea24fd78f77c6108c

## Contents

vscDebugger-package . . . . .	2
.vsc.applyVarInfos . . . . .	2
.vsc.debugSource . . . . .	4
.vsc.getAllVarInfoFields . . . . .	4
.vsc.load_all . . . . .	5
.vsc.onError . . . . .	6
.vsc.refreshBreakpoints . . . . .	6
outputOverwrites . . . . .	6

---

vscDebugger-package    *vscDebugger*

---

### Description

(Partial) implementation of the [Debug Adapter Protocol](#) for R.

### Details

Provides support for the vscode extension [vscode-R-Debugger](#).

### Author(s)

**Maintainer:** Manuel Hentschel <Manuel.Hentschel@outlook.com>

Other contributors:

- Kun Ren <mail@renkun.me> [contributor]
- Denes Toth <toth.denes@kogentum.hu> ([ORCID](#)) [contributor]

### See Also

Useful links:

- <https://ManuelHentschel.github.io/vscDebugger/>
- <https://github.com/ManuelHentschel/vscDebugger>
- Report bugs at <https://github.com/ManuelHentschel/vscDebugger/issues>

---

.vsc.applyVarInfos    *Get info about a variable*

---

### Description

Get info about a variable by applying the relevant varInfo entries.

### Usage

```
.vsc.applyVarInfos(  
  v,  
  infos = character(0),  
  stackingInfos = character(0),  
  verbose = getOption("vsc.verboseVarInfos", FALSE),  
  ind = NULL  
)
```

### Arguments

<code>v</code>	The variable, any R value.
<code>infos</code>	Character vector of field names in <code>varInfo</code> . Retrieve first match.
<code>stackingInfos</code>	Character vector of field names in <code>varInfo</code> . Retrieve all matches.
<code>verbose</code>	Whether to print debug info using <code>logCat</code> , <code>logPrint</code> .
<code>ind</code>	The indices to retrieve if child variables are retrieved.

### Details

The allowed `varInfo` entries and their types are:

- `childVars`: `MinimalVariable[]`
- `nChildVars`: number
- `customAttributes`: `MinimalVariable[]`
- `internalAttributes`: `MinimalVariable[]`
- `toString`: string
- `type`: string
- `evaluateName`: string
- `printFunc`: function | boolean

Where `MinimalVariable[]` refers to a list of named lists with entries:

- `name`: string
- `rValue`: any R value
- `setter`: (optional, undocumented)
- `setInfo`: (optional, undocumented)

### Value

A named list, containing the corresponding `varInfo` results.

### See Also

[varInfos](#)

---

`.vsc.debugSource`      *Modified version of `base::source`*

---

### Description

Modified version of `base::source` that honors breakpoints set by the debugger.

### Usage

```
.vsc.debugSource(
  file,
  local = FALSE,
  envir = NULL,
  chdir = FALSE,
  print.eval = NULL,
  encoding = "unknown",
  ...
)
```

### Arguments

<code>file</code>	String giving the name of the file to be sourced. Connections etc. are not supported!
<code>local</code>	Same as in <a href="#">base::source()</a> , can be overwritten by specifying an environment in <code>envir</code> .
<code>envir</code>	The environment in which to evaluate the sourced code. Overwrites <code>local</code> , if specified.
<code>chdir</code>	Whether to temporarily change the working directory to the location of <code>file</code> .
<code>encoding</code>	The encoding to be used by <a href="#">base::parse()</a> .
<code>...</code>	Further arguments are ignored but allowed for compatibility with <a href="#">base::source()</a> .

### See Also

Other overwrites: [.vsc.load\\_all\(\)](#), [outputOverwrites](#)

---

`.vsc.getAllVarInfoFields`  
*Tools to modify/debug varInfos*

---

### Description

Tools to check the `varInfos` computed for variables and modify the list of `varInfos` used.

**Usage**

```
.vsc.getAllVarInfoFields(varInfos = session$varInfos)
.vsc.applyAllVarInfos(v, verbose = TRUE)
.vsc.resetVarInfos()
.vsc.addVarInfo(varInfo, overwrite = FALSE)
.vsc.removeVarInfos(positions)
.vsc.getVarInfos(positions = NULL)
.vsc.checkVarInfos(varInfos = session$varInfos)
```

**Arguments**

<code>varInfos</code>	List of <code>varInfos</code>
<code>v</code>	Variable, any R object
<code>verbose</code>	passed to <code>.vsc.applyVarInfos</code>
<code>varInfo</code>	A <code>varInfo</code> , i.e. named list.
<code>overwrite</code>	Boolean, whether to overwrite the entry in that position
<code>positions</code>	Numeric or character vector, the entries to retrieve/remove

**See Also**

[.vsc.applyVarInfos](#)

---

`.vsc.load_all`      *Modified version of* `pkgload::load_all()`

---

**Description**

Modified version of `pkgload::load_all()`

**Usage**

```
.vsc.load_all(...)
```

**See Also**

Other overwrites: [.vsc.debugSource\(\)](#), [outputOverwrites](#)

---

`.vsc.onError`                      *Error handler*

---

### Description

Error handler used by vsc. Set with `options(error = .vsc.onError)`

### Usage

```
.vsc.onError(err = NULL)
```

### Arguments

`err`                      The message to be sent to vsc. Defaults to `geterrmessage()`

---

`.vsc.refreshBreakpoints`  
    *Refresh Breakpoints*

---

### Description

Refresh breakpoints known to the debugger Can be used if breakpoints were invalidated by e.g. `load_all()` or `source()`

### Usage

```
.vsc.refreshBreakpoints(envs = NULL)
```

---

`outputOverwrites`                      *Modified Output Functions*

---

### Description

Modified versions of core R functions that add debugger specific functionality. With the default debug configuration, the normal R functions are overwritten with these (using `attach()`).

The modified functions are designed to be mostly interchangeable with their counterparts. If the overwrites are not required, they can be toggled using the launch config entries `overwriteCat`, `overwritePrint`, `overwriteStr`, and `overwriteMessage`.

**Usage**

```
.vsc.cat(..., skipCalls = 0, showSource = TRUE)

.vsc.print(x, ..., skipCalls = 0, showSource = TRUE)

.vsc.message(
  ...,
  domain = NULL,
  appendLF = TRUE,
  showSource = TRUE,
  skipCalls = 0
)

.vsc.str(object, ..., skipCalls = 0, showSource = TRUE)
```

**Arguments**

...	Further arguments that are passed to the original function
skipCalls	The number of calls to skip when reporting the calling file and line. Can be useful e.g. inside a log function.
showSource	Whether to show the calling source file and line
x	Same as in <a href="#">base::print()</a>
domain	Same as in <a href="#">base::message()</a>
appendLF	Same as in <a href="#">base::message()</a>
object	Same as in <a href="#">utils::str()</a>

**Value**

These functions return the same value as the overwritten functions themselves.

**See Also**

[base::cat\(\)](#)

[base::print\(\)](#)

[base::message\(\)](#)

[utils::str\(\)](#)

Other overwrites: [.vsc.debugSource\(\)](#), [.vsc.load\\_all\(\)](#)

# Index

## \* **overwrites**

- .vsc.debugSource, 4
- .vsc.load\_all, 5
- outputOverwrites, 6
- .vsc.addVarInfo
  - (.vsc.getAllVarInfoFields), 4
- .vsc.applyAllVarInfos
  - (.vsc.getAllVarInfoFields), 4
- .vsc.applyVarInfos, 2, 5
- .vsc.cat (outputOverwrites), 6
- .vsc.checkVarInfos
  - (.vsc.getAllVarInfoFields), 4
- .vsc.debugSource, 4, 5, 7
- .vsc.getAllVarInfoFields, 4
- .vsc.getVarInfos
  - (.vsc.getAllVarInfoFields), 4
- .vsc.load\_all, 4, 5, 7
- .vsc.message (outputOverwrites), 6
- .vsc.onError, 6
- .vsc.print (outputOverwrites), 6
- .vsc.refreshBreakpoints, 6
- .vsc.removeVarInfos
  - (.vsc.getAllVarInfoFields), 4
- .vsc.resetVarInfos
  - (.vsc.getAllVarInfoFields), 4
- .vsc.str (outputOverwrites), 6

attach(), 6

- base::cat(), 7
- base::message(), 7
- base::parse(), 4
- base::print(), 7
- base::source(), 4

outputOverwrites, 4, 5, 6

utils::str(), 7

varInfos, 3

vscDebugger (vscDebugger-package), 2

vscDebugger-package, 2