

# Package: s7contract (via r-universe)

July 13, 2026

**Title** 'Go'-Like Interfaces and 'Rust'-Like Traits with 'S7'

**Version** 0.1.0.9000

**Description** Contract helpers built with 'S7' for expressing runtime protocols around ordinary 'S7' dispatch. Structural interfaces describe small sets of required 'S7' generics, while explicit traits record registered implementations with optional default methods and associated metadata. Optional runtime checks can validate argument and return specifications in contract-scoped evaluation.

**License** GPL (>= 3)

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3

**Depends** R (>= 4.3.0)

**Imports** S7

**Suggests** knitr, rmarkdown, tinytest

**VignetteBuilder** knitr

**URL** <https://github.com/abiomix/abiomix-r-open>,  
<https://github.com/sounkou-bioinfo/s7contract>,  
<https://sounkou-bioinfo.github.io/s7contract/>

**BugReports** <https://github.com/abiomix/abiomix-r-open/issues>

**Config/abiomix/public-license** GPL-3.0-or-later

**Config/abiomix/commercial-license** Separate written agreement for  
Abiomix-owned contributions

**Config/abiomix/license-policy** <https://github.com/abiomix/abiomix-r-open/blob/main/LICENSES.md>

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

**Date/Publication** 2026-07-13 08:37:42 UTC

**RemoteUrl** <https://github.com/abiomix/abiomix-r-open>

**RemoteRef** HEAD

**RemoteSha** dd82290ba296385fcc7d7dfed339886e9deb41a2  
**RemoteSubdir** packages/s7contract

Contents

|                        |           |
|------------------------|-----------|
| %::%                   | 2         |
| interface_requirements | 3         |
| new_interface          | 4         |
| new_trait              | 5         |
| s7contract             | 7         |
| trait_methods          | 8         |
| <b>Index</b>           | <b>10</b> |

---

|      |                                                                 |
|------|-----------------------------------------------------------------|
| %::% | <i>Evaluate an S7 call under an interface or trait contract</i> |
|------|-----------------------------------------------------------------|

---

Description

with(contract, expr) and expr %::% contract evaluate expr in a contract mask. Required generics are shadowed by checking wrappers, so calls to those generics use normal S7 dispatch while checking the optional argument and return specifications stored in an interface requirement or trait method.

Usage

expr %::% contract

Arguments

- expr           An expression evaluated in a contract mask. Calls to generics named in the contract are checked.
- contract       An interface created by `new_interface()` or a trait created by `new_trait()`.

Value

The value of expr, after any optional return check.

Examples

```
local({
  draw <- S7::new_generic("draw", "x", function(x, color) {
    S7::S7_dispatch()
  })
  Circle <- S7::new_class("TypedCircle", properties = list(r = S7::class_double))
  S7::method(draw, Circle) <- function(x, color) paste(color, x@r)
  Drawable <- new_interface(
    "TypedDrawable",
```

```

    generics = list(draw = interface_requirement(
      draw,
      args = list(color = S7::class_character),
      returns = S7::class_character
    ))
  )
  with(Drawable, draw(Circle(r = 2), color = "red"))
  checked_draw <- with(Drawable, function(x) draw(x, color = "red"))
  checked_draw(Circle(r = 2))
  draw(Circle(r = 2), color = "red") %::% Drawable
})

```

---

interface\_requirements

*Inspect or check a Go-like structural interface*


---

## Description

Inspect or check a Go-like structural interface

## Usage

```

interface_requirements(interface, inherited = TRUE)

interface_report(x, interface)

missing_requirements(x, interface)

implements(x, interface)

assert_implements(x, interface, arg = deparse(substitute(x)))

as_interface(x, interface)

```

## Arguments

|           |                                                        |
|-----------|--------------------------------------------------------|
| interface | An interface created by new_interface().               |
| inherited | Include inherited requirements from parent interfaces? |
| x         | An object, or an S7 class/base class wrapper.          |
| arg       | Name to use in error messages.                         |

## Value

interface\_requirements() returns a named list of interface\_requirement() objects. interface\_report() and missing\_requirements() return data frames. implements() returns a single logical value. assert\_implements() and as\_interface() return x, unchanged.

new\_interface

*Build a Go-like structural interface on top of S7***Description**

`new_interface()` models the method-list part of Go interfaces as a list of required S7 generics. An interface is just a named set of required generics, and a class or object satisfies it when S7 can find a method for every required generic.

**Usage**

```
new_interface(
    name,
    generics = list(),
    parents = list(),
    package = NULL,
    methods = NULL
)

interface_requirement(
    generic,
    name = NULL,
    args = list(),
    returns = S7::class_any
)
```

**Arguments**

|          |                                                                                                                                                                                                                                                                                                                                                                           |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name     | For <code>new_interface()</code> , the interface name. For <code>interface_requirement()</code> , the requirement name; it defaults to the generic name when omitted.                                                                                                                                                                                                     |
| generics | For <code>new_interface()</code> , a named list of S7 generics or <code>interface_requirement()</code> objects. These are generic functions because S7 methods are registered separately on generics.                                                                                                                                                                     |
| parents  | Optional interface or list of interfaces to embed.                                                                                                                                                                                                                                                                                                                        |
| package  | Optional package name used only for display.                                                                                                                                                                                                                                                                                                                              |
| methods  | Compatibility alias for generics.                                                                                                                                                                                                                                                                                                                                         |
| generic  | An S7 generic function.                                                                                                                                                                                                                                                                                                                                                   |
| args     | Optional named list of S7 classes, interfaces, or traits for runtime argument checking with <code>with()</code> or <code>%::%</code> . Arguments named in <code>args</code> are also checked against generic and method formals during conformance checks. Dispatch arguments other than the first can use S7 classes or unions to refine multiple-dispatch requirements. |
| returns  | Optional S7 class, interface, or trait for runtime return checking with <code>with()</code> or <code>%::%</code> ; defaults to <code>S7::class_any</code> .                                                                                                                                                                                                               |

## Details

This deliberately mirrors Go's basic interfaces defined only by methods. The intended style is to define small interfaces at the point where consuming code needs a behavior, not beside a single concrete implementation. Define S7 classes, generics, and methods normally; then let consumers name the protocol they accept. Up-front interfaces can still be useful for deliberate package protocols, abstract data types, or recursive protocols.

It does not attempt to emulate Go's full post-1.18 type-set language such as tilde type terms, unions of concrete types, or pointer/value receiver rules.

## Value

`new_interface()` returns an S7 object of class `s7_interface`. `interface_requirement()` returns an S7 object of class `s7_interface_requirement`.

## Examples

```
local({
  area <- S7::new_generic("area", "x")
  draw <- S7::new_generic("draw", "x")

  Circle <- S7::new_class(
    "Circle",
    properties = list(r = S7::class_double)
  )
  Rect <- S7::new_class(
    "Rect",
    properties = list(w = S7::class_double, h = S7::class_double)
  )

  S7::method(area, Circle) <- function(x) pi * x@r^2
  S7::method(draw, Circle) <- function(x) sprintf("circle(r = %s)", x@r)
  S7::method(area, Rect) <- function(x) x@w * x@h

  Drawable <- new_interface("Drawable", generics = list(draw = draw))
  Shape <- new_interface("Shape", generics = list(area = area), parents = Drawable)

  implements(Circle, Shape)
  missing_requirements(Rect, Shape)
})
```

## Description

`new_trait()` adds a nominal contract registry on top of S7 dispatch. A class only has the trait after `impl_trait()` records the implementation, even if compatible S7 methods already exist.

**Usage**

```

new_trait(
  name,
  methods = list(),
  parents = list(),
  assoc_types = character(),
  assoc_consts = list(),
  package = NULL
)

trait_method(
  generic,
  default = NULL,
  name = NULL,
  args = list(),
  returns = S7::class_any
)

```

**Arguments**

|              |                                                                                                                                                                                                                                                       |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| name         | For <code>new_trait()</code> , the trait name. For <code>trait_method()</code> , the method name; it defaults to the generic name when omitted.                                                                                                       |
| methods      | For <code>new_trait()</code> , a named list of S7 generics or <code>trait_method()</code> objects.                                                                                                                                                    |
| parents      | Optional trait or list of supertraits.                                                                                                                                                                                                                |
| assoc_types  | Required associated type names, or a named list of default associated type values.                                                                                                                                                                    |
| assoc_consts | Required associated constant names, or a named list of default constant values.                                                                                                                                                                       |
| package      | Optional package name used only for display.                                                                                                                                                                                                          |
| generic      | An S7 generic function.                                                                                                                                                                                                                               |
| default      | Optional default implementation. If supplied, <code>impl_trait()</code> uses it when a class does not provide an override for that method.                                                                                                            |
| args         | Optional named list of S7 classes, interfaces, or traits for runtime argument checking with <code>with()</code> or <code>%::%</code> . Dispatch arguments other than the first can use S7 classes or unions to refine multiple-dispatch requirements. |
| returns      | Optional S7 class, interface, or trait for runtime return checking with <code>with()</code> or <code>%::%</code> ; defaults to <code>S7::class_any</code> .                                                                                           |

**Details**

This makes default methods and associated metadata practical, but the result remains a runtime R abstraction. It does not emulate Rust's compile-time trait bounds, coherence, orphan rules, or type-checked associated types.

**Value**

`new_trait()` returns an S7 object of class `s7_trait`. `trait_method()` returns an S7 object of class `s7_trait_method`.

## Examples

```
local({
  area <- S7::new_generic("area", "x")
  perimeter <- S7::new_generic("perimeter", "x")

  Circle <- S7::new_class(
    "Circle",
    properties = list(r = S7::class_double)
  )

  Measurable <- new_trait(
    "Measurable",
    methods = list(
      area = trait_method(area),
      perimeter = trait_method(perimeter, default = function(x) NA_real_)
    ),
    assoc_consts = c("UNITS")
  )

  impl_trait(
    Measurable,
    Circle,
    methods = list(area = function(x) pi * x@r^2),
    assoc_consts = list(UNITS = "unitless")
  )

  has_trait(Circle, Measurable)
  trait_call(Measurable, "area", Circle(r = 2))
  trait_assoc_const(Measurable, Circle, "UNITS")
})
```

---

s7contract

s7contract: Contract Helpers for S7

---

## Description

s7contract provides two experimental contract layers on top of S7:

## Details

- Go-like structural interfaces defined by required generics.
- Rust-like explicit traits with default methods and associated metadata.

The package keeps actual method dispatch inside ordinary S7 generics and uses runtime checks to describe or assert conformance.

## Author(s)

**Maintainer:** Soukoku Mahamane Toure <soukoutoure@gmail.com>

**See Also**

Useful links:

- <https://github.com/souunkou-bioinfo/s7contract>
- <https://souunkou-bioinfo.github.io/s7contract/>
- Report bugs at <https://github.com/souunkou-bioinfo/s7contract/issues>

---

trait\_methods

*Inspect or use a Rust-like explicit trait*


---

**Description**

Inspect or use a Rust-like explicit trait

**Usage**

```
trait_methods(trait, inherited = TRUE)

impl_trait(
  trait,
  class,
  methods = list(),
  assoc_types = list(),
  assoc_consts = list(),
  replace = FALSE
)

trait_report(x, trait)

has_trait(x, trait)

assert_trait(x, trait, arg = deparse(substitute(x)))

trait_call(trait, method, x, ...)

trait_assoc_type(trait, x, name)

trait_assoc_const(trait, x, name)
```

**Arguments**

|           |                                                                                                                     |
|-----------|---------------------------------------------------------------------------------------------------------------------|
| trait     | A trait created by new_trait().                                                                                     |
| inherited | Include inherited methods from supertraits?                                                                         |
| class     | An S7 class or base class wrapper.                                                                                  |
| methods   | Named list of method implementations. Omitted trait methods use their default implementation when one is available. |



|              |                                                                                          |
|--------------|------------------------------------------------------------------------------------------|
| assoc_types  | Named list of associated type values.                                                    |
| assoc_consts | Named list of associated constant values.                                                |
| replace      | Replace an existing implementation record and silence warnings about visible S7 methods? |
| x            | An object or class.                                                                      |
| arg          | Name to use in error messages.                                                           |
| method       | Method name within the trait.                                                            |
| ...          | Additional arguments passed to the S7 generic.                                           |
| name         | Associated item name.                                                                    |

**Value**

`trait_methods()` returns a named list of `trait_method()` objects. `impl_trait()` returns the stored implementation record, invisibly. `trait_report()` returns a one-row data frame. `has_trait()` returns a single logical value. `assert_trait()` returns `x`, unchanged. `trait_call()` returns the result of the underlying S7 generic. `trait_assoc_type()` and `trait_assoc_const()` return the stored associated item value.

# Index

`%.: %`, 2

`as_interface (interface_requirements)`, 3

`assert_implements`  
    `(interface_requirements)`, 3

`assert_trait (trait_methods)`, 8

`contract_syntax (%.: %)`, 2

`has_trait (trait_methods)`, 8

`impl_trait (trait_methods)`, 8

`implements (interface_requirements)`, 3

`interface_report`  
    `(interface_requirements)`, 3

`interface_requirement (new_interface)`, 4

`interface_requirements`, 3

`missing_requirements`  
    `(interface_requirements)`, 3

`new_interface`, 4

`new_interface()`, 2

`new_trait`, 5

`new_trait()`, 2

`s7contract`, 7

`s7contract-package (s7contract)`, 7

`trait_assoc_const (trait_methods)`, 8

`trait_assoc_type (trait_methods)`, 8

`trait_call (trait_methods)`, 8

`trait_method (new_trait)`, 5

`trait_methods`, 8

`trait_report (trait_methods)`, 8