# Package 'fhircrackr'

March 21, 2024

**Type** Package

**Title** Handling HL7 FHIR® Resources in R

**Version** 2.2.0

**Date** 2024-03-21

**Description** Useful tools for conveniently downloading FHIR resources in xml format and converting them to R data.frames. The package uses FHIR-search to download bundles from a FHIR server, provides functions to save and read xml-files containing such bundles and allows flattening the bundles to data.frames using XPath expressions. FHIR® is the registered trademark of HL7 and is used with the permission of HL7. Use of the FHIR trademark does not constitute endorsement of this product by HL7.

**BugReports** https://github.com/POLAR-fhiR/fhircrackr/issues

**License** GPL-3

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.2.1

**Imports** xml2, stringr, httr, utils, data.table, methods, parallel, lifecycle

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Depends** R (>= 4.1.0)

**Collate** 'fhir_body.R' 'fhir_resource.R' 'fhir_url.R' 'fhir_bundle.R' 'fhir_bundle_list.R' 'build_resources.R' 'design.R' 'download_resources.R' 'fhir_xpath_expression.R' 'fhir_columns.R' 'fhir_resource_type.R' 'fhir_table_description.R' 'fhir_design.R' 'fhir_style.R' 'fhir_table_list.R' 'fhir_tree.R' 'flatten_resources.R' 'miscellaneous.R' 'multiple_entries.R' 'sample_resources.R'

**NeedsCompilation** no

**Author** Thomas Peschel [aut],
Julia Palm [aut, cre] (<https://orcid.org/0000-0003-1568-5893>),
Jens Przybilla [aut],
Frank Meineke [aut] (<https://orcid.org/0000-0002-9256-7543>)

**Maintainer** Julia Palm <julia.palm@med.uni-jena.de>

**Repository** CRAN

**Date/Publication** 2024-03-21 14:40:06 UTC

# R topics documented:

as_fhir . . . 3
example_bundles1 . . . 4
fhir_authenticate . . . 12
fhir_body . . . 13
fhir_body-class . . . 14
fhir_build_bundle . . . 14
fhir_build_resource . . . 17
fhir_bundle-class . . . 18
fhir_bundle_list . . . 19
fhir_bundle_list-class . . . 20
fhir_bundle_serialized-class . . . 20
fhir_bundle_xml . . . 20
fhir_bundle_xml-class . . . 21
fhir_canonical_design . . . 21
fhir_capability_statement . . . 22
fhir_cast . . . 23
fhir_columns . . . 25
fhir_columns-class . . . 26
fhir_common_columns . . . 26
fhir_count_resource . . . 27
fhir_crack . . . 29
fhir_current_request . . . 32
fhir_design . . . 33
fhir_design-class . . . 36
fhir_df_list-class . . . 36
fhir_dt_list-class . . . 36
fhir_get_resources_by_ids . . . 37
fhir_get_resource_ids . . . 39
fhir_load . . . 40
fhir_load_design . . . 41
fhir_melt . . . 43
fhir_next_bundle_url . . . 44
fhir_post . . . 46
fhir_put . . . 48
fhir_recent_http_error . . . 50
fhir_request . . . 51
fhir_resource-class . . . 52

fhir_resource_serialized-class . . . 52
fhir_resource_type . . . 52
fhir_resource_type-class . . . 53
fhir_resource_xml . . . 53
fhir_resource_xml-class . . . 54
fhir_rm_div . . . 54
fhir_rm_indices . . . 55
fhir_rm_tag . . . 56
fhir_sample_resources . . . 58
fhir_sample_resources_by_ids . . . 60
fhir_save . . . 63
fhir_save_design . . . 63
fhir_search . . . 64
fhir_serialize . . . 68
fhir_table_description . . . 69
fhir_table_description-class . . . 72
fhir_tree . . . 74
fhir_unserialize . . . 76
fhir_url . . . 77
fhir_url-class . . . 79
fhir_xpath_expression . . . 79
fhir_xpath_expression-class . . . 79
medication_bundles . . . 80
pastep . . . 81
paste_paths . . . 82
transaction_bundle_example . . . 82

**Index** **86**

---

`as_fhir` *Coerce character vector to fhir_bundle_list*

---

## Description

Tries to convert a character vector containing xml strings representing FHIR bundles to an object of class fhir_bundle_list.

## Usage

```
as_fhir(x)
```

## Arguments

| | |
|---|---|
| `x` | A character vector where each element is a string representing an xml FHIR bundle. |

## Examples

```
#character vector containing fhir bundles
bundle_strings <- c(
"<Bundle>
 <type value='searchset'/>
 <entry>
  <resource>
    <Patient>
       <id value='id1'/>
      <name>
         <given value='Marie'/>
      </name>
    </Patient>
  </resource>
 </entry>
</Bundle>",
"<Bundle>
 <type value='searchset'/>
 <entry>
  <resource>
    <Patient>
       <id value='id2'/>
      <name>
         <given value='Max'/>
      </name>
    </Patient>
  </resource>
 </entry>
</Bundle>"
)

#convert to FHIR bundle list
bundles <- as_fhir(bundle_strings)
```

---

`example_bundles1` *Toy example bundles for multiple entries*

---

## Description

These data examples are bundles that contain very few, very simple resources that have multiple entries and can be used for demonstration purposes. See **Source** for how the xml versions look.

## Usage

```
example_bundles1

example_bundles2
```

example_bundles3

example_bundles4

example_bundles5

## Format

An object of class fhir_bundle_list of length 1.

An object of class fhir_bundle_list of length 1.

An object of class fhir_bundle_list of length 1.

An object of class fhir_bundle_list of length 1.

An object of class fhir_bundle_list of length 1.

## Details

example_bundles1 contains 1 bundle with 2 Patient resources.

example_bundles2 contains 1 bundle with 3 Patient resources.

example_bundles3 contains 1 bundle with 3 Patient resources and 1 Observation resource.

example_bundles4 contains 1 bundle with 2 Medication resources, one of which has some @id xml attributes

example_bundles5 contains 1 bundle with 2 Observation resources.

## Source

**example_bundles1**

```
<Bundle>
 <type value='searchset'/>
 <entry>
  <resource>
    <Patient>
       <id value='id1'/>
      <address>
         <use value='home'/>
         <city value='Amsterdam'/>
         <type value='physical'/>
         <country value='Netherlands'/>
      </address>
      <name>
         <given value='Marie'/>
      </name>
    </Patient>
  </resource>
 </entry>
```

```
 <entry>
  <resource>
    <Patient>
       <id value='id3'/>
      <address>
         <use value='home'/>
         <city value='Berlin'/>
      </address>
       <address>
          <type value='postal'/>
          <country value='France'/>
       </address>
      <address>
         <use value='work'/>
         <city value='London'/>
         <type value='postal'/>
         <country value='England'/>
      </address>
       <name>
          <given value='Frank'/>
       </name>
       <name>
          <given value='Max'/>
       </name>
    </Patient>
  </resource>
 </entry>
</Bundle>
```

**example_bundles2**

```
<Bundle>
 <type value='searchset'/>
 <entry>
  <resource>
     <Patient>
        <id value='id1'/>
        <address>
           <use value='home'/>
           <city value='Amsterdam'/>
           <type value='physical'/>
           <country value='Netherlands'/>
        </address>
        <name>
           <given value='Marie'/>
        </name>
     </Patient>
   </resource>
 </entry>
```

```
<entry>
 <resource>
   <Patient>
       <id value='id2'/>
       <address>
          <use value='home'/>
          <city value='Rome'/>
          <type value='physical'/>
          <country value='Italy'/>
       </address>
       <address>
          <use value='work'/>
          <city value='Stockholm'/>
         <type value='postal'/>
          <country value='Sweden'/>
       </address>
       <name>
          <given value='Susie'/>
      </name>
   </Patient>
 </resource>
</entry>

<entry>
 <resource>
   <Patient>
      <id value='id3'/>
      <address>
         <use value='home'/>
         <city value='Berlin'/>
      </address>
      <address>
         <type value='postal'/>
         <country value='France'/>
      </address>
      <address>
         <use value='work'/>
         <city value='London'/>
         <type value='postal'/>
         <country value='England'/>
      </address>
      <name>
         <given value='Frank'/>
     </name>
     <name>
         <given value='Max'/>
      </name>
```

```
    </Patient>
  </resource>
 </entry>
</Bundle>
```

**example_bundles3**

```
<Bundle>
 <type value='searchset'/>
 <entry>
  <resource>
    <Patient>
       <id value='id1'/>
       <address>
          <use value='home'/>
          <city value='Amsterdam'/>
          <type value='physical'/>
          <country value='Netherlands'/>
       </address>
       <name>
          <given value='Marie'/>
       </name>
    </Patient>
  </resource>
 </entry>

 <entry>
  <resource>
    <Patient>
       <id value='id2'/>
       <address>
          <use value='home'/>
          <city value='Rome'/>
          <type value='physical'/>
          <country value='Italy'/>
       </address>
       <address>
          <use value='work'/>
          <city value='Stockholm'/>
          <type value='postal'/>
          <country value='Sweden'/>
       </address>
       <name>
          <given value='Susie'/>
       </name>
    </Patient>
  </resource>
 </entry>
```

```
<entry>
 <resource>
   <Patient>
      <id value='id3'/>
      <address>
         <use value='home'/>
         <city value='Berlin'/>
      </address>
      <address>
         <type value='postal'/>
         <country value='France'/>
      </address>
      <address>
         <use value='work'/>
         <city value='London'/>
         <type value='postal'/>
         <country value='England'/>
      </address>
      <name>
         <given value='Frank'/>
      </name>
      <name>
         <given value='Max'/>
      </name>
   </Patient>
 <resource/>
</entry>

<entry>
 <resource>
   <Observation>
      <id value = 'obs1'/>
      <code>
         <coding>
            <system value='http://loinc.org'/>
            <code value='29463-7'/>
            <display value='Body Weight'/>
         </coding>
         <coding>
            <system value='http://snomed.info/sct'/>
            <code value='27113001'/>
            <display value='Body weight'/>
         </coding>
      </code>
      <subject>
         <reference value='Patient/id3'/>
      </subject>
   </Observation>
```

```
  </resource>
 </entry>
</Bundle>"
```

**example_bundles4**

```
<Bundle>
    <type value='searchset'/>
 <entry>
 <resource>
<Medication>
    <id value='1285'/>
    <code>
        <coding>
            <system value='http://www.nlm.nih.gov/research/umls/rxnorm'/>
            <code value='1594660'/>
            <display value='Alemtuzumab 10mg/ml (Lemtrada)'/>
        </coding>
    </code>
    <ingredient id='1'>
     <itemReference>
         <reference value='Substance/5463'/>
     </itemReference>
    </ingredient>
    <ingredient id='2'>
     <itemReference>
         <reference value='Substance/3401'/>
     </itemReference>
    </ingredient>
</Medication>
</resource>
 </entry>

 <entry>
 <resource>
        <Medication>
            <id value='45226'/>
            <code>
                <coding>
                    <system value='http://snomed.info/sct'/>
                    <code value='373994007'/>
                    <display value='Prednisone 5mg tablet (Product)'/>
                </coding>
                <text value='Prednisone 5mg tablet (Product)'/>
            </code>
   <ingredient id='1'>
     <itemReference>
         <reference value='Substance/6912'/>
```

```
     </itemReference>
    </ingredient>
    <ingredient id='2'>
     <itemReference>
         <reference value='Substance/3710'/>
     </itemReference>
    </ingredient>
</Medication>
</resource>
 </entry>

</Bundle>
```

**example_bundles5**

```
<Bundle>
 <type value='searchset'/>
 <entry>
  <resource>
    <Observation>
       <id value = 'obs1'/>
       <code>
          <coding>
             <system value='http://loinc.org'/>
             <code value='29463-7'/>
             <display value='Body Weight'/>
          </coding>
          <coding>
             <system value='http://snomed.info/sct'/>
             <code value='27113001'/>
             <display value='Body weight'/>
          </coding>
       </code>
       <subject>
          <reference value='Patient/id3'/>
       </subject>
    </Observation>
  </resource>
 </entry>

 <entry>
  <resource>
    <Observation>
       <id value = 'obs2'/>
       <code>
          <coding>
             <system value='http://loinc.org'/>
             <code value='8302-2'/>
```

```
            <display value='Body Height'/>
         </coding>
         <coding>
            <system value='http://snomed.info/sct'/>
            <code value='50373000'/>
            <display value='Body height measure'/>
         </coding>
      </code>
      <subject>
         <reference value='Patient/id3'/>
      </subject>
    </Observation>
  </resource>
 </entry>
</Bundle>"
```

## Examples

```
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles1)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles2)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles3)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles4)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = example_bundles5)
```

---

# fhir_authenticate *Create token for Authentication*

---

## Description

This function is a wrapper to create an httr::Token object for authentication with OAuth2/OpenID Connect. Internally, it calls `httr::oauth_app()`, `httr::oauth_endpoint()` and `httr::oauth2.0_token()` to create a token that can then be used in fhir_search.

## Usage

```
fhir_authenticate(
  secret,
  key,
  base_url,
  access,
  authorize,
  query_authorize_extra = list()
)
```

### Arguments

| | |
|---|---|
| secret | The consumer/client secret, belonging to key. |
| key | Consumer key, also called client ID. For Keycloak this would for instance be the Keycloak client, e.g. "postman". |
| base_url | The URL the user will be redirected to after authorization is complete. This will usually be the base url of you FHIR server. |
| access | The url used to exchange unauthenticated for authenticated token. This can be identical to authorize. |
| authorize | The url to send the client for authorization. |
| query_authorize_extra | A named list holding query parameters to append to initial auth page query. Could hold info about user identity and scope for keycloak like this: |

```
list(scope = "openid",
     grant_type = "password",
     username = "fhir-user",
     password = "fhirtest")
```

---

## fhir_body *Create fhir_body object*

---

### Description

Create fhir_body object

### Usage

```
fhir_body(content, type)

## S4 method for signature 'list,missing'
fhir_body(content)

## S4 method for signature 'list,character'
fhir_body(content, type)

## S4 method for signature 'character,character'
fhir_body(content, type)
```

### Arguments

| | |
|---|---|
| content | A character vector of length one representing the body for the post in the format specified in type. If you provide a named list here, it will be taken as key value pairs of FHIR search parameters and will be concatenated appropriately. In this case the type will automatically be set to "application/x-www-form-urlencoded". See examples. |
| type | A string defining the type of the body e.g. "application/x-www-form-urlencoded" or "xml". |

## Value

An object of type fhir_body.

## Examples

```
#body that could be used in a FHIR search request POSTed to an URL like baseurl/Patient/_search
fhir_body(content = "gender=female&_summary=count", type="application/x-www-form-urlencoded")
fhir_body(content = list("gender" = "female", "_summary" = "count"))
```

---

# fhir_body-class *An s4 class to represent a body for a POST to a FHIR server*

---

## Description

Objects of this class should always be created with a call to the function `fhir_body()`

## Slots

`content` A vector of length one representing the body for the post.

`type` A vector of length one defining the type of the body e.g. `"application/x-www-form-urlencoded"` or `"xml"`.

---

# fhir_build_bundle *Build a FHIR bundle*

---

## Description

This function takes a table as produced by `fhir_crack()` with `format="wide"` and builds a fhir_bundle_xml object from it. It is primarily used to create transaction/batch bundles to POST back to a FHIR server. The column names of the table must represent the XPath expression of the respective element with indices for repeating items. A table like this is produced when FHIR resources have been cracked with `fhir_crack()` without assigning explicit column names in the fhir_design/fhir_table_description and the format has been set to `"wide"`.

## Usage

```
fhir_build_bundle(
  table,
  brackets,
  resource_type,
  bundle_type = "transaction",
  verbose = 1
)

## S4 method for signature 'data.frame'
```

```
fhir_build_bundle(
  table,
  brackets,
  resource_type,
  bundle_type = "transaction",
  verbose = 1
)

## S4 method for signature 'list'
fhir_build_bundle(table, brackets, bundle_type = "transaction", verbose = 1)
```

## Arguments

| | |
|---|---|
| table | A wide table as produced by fhir_crack(), possibly modified (see details) or a named list of wide tables, if different resource types have to be included in the same bundle. In this case the names of the list elements must correspond to the resource type represented in the table! |
| brackets | A character vector of length one. The brackets used for cracking. |
| resource_type | A character vector of length one or fhir_resource_type object indicating which resource type is represented in the table, if a single table is provided. This argument is ignored when table is a named list of tables. |
| bundle_type | A character vector of length one defining the bundle type. Will usually be either "transaction" (the default) or "batch". |
| verbose | An integer vector of length one. If 0, nothing is printed, if > 0 progress message is printed. Defaults to 1. |

## Details

The typical use case would look like this:

1. Download resources from a server with fhir_search()
2. Crack to wide format them with fhir_crack()
3. Do something to values (e.g. some kind of anonymization)
4. Translate the data back into FHIR resources with fhir_build_bundle()
5. Post the resources to a server

A FHIR bundle that can be POSTed to a server is usually of type transaction or batch. Each entry of these bundles consists of the resource itself as well as an instruction for the server of what to to with the resource. A very simple example looks like this:

```
<Bundle>
   <type value="transaction"/>
  <entry>
     <resource>
        <Patient>
           <id value="id1"/>
       <address>
```

```
          <city value="Amsterdam"/>
          <country value="Netherlands"/>
       </address>
       <name>
          <given value="Marie"/>
       </name>
        </Patient>
   </resource>
   <request>
  <method value="POST"/>
  <url value="Patient"/>
   </request>
</entry>
<entry>
    <resource>
       <Patient>
          <id value="id2"/>
      <address>
         <city value="Paris"/>
         <country value="France"/>
      </address>
      <name>
         <given value="Anne"/>
      </name>
       </Patient>
   </resource>
   <request>
  <method value="POST"/>
  <url value="Patient"/>
   </request>
</entry>
</Bundle>
```

In this example the bundle contains two Patient resources that are sent to server with a POST. For more information the structure of transaction/batch bundles, please see the FHIR documentation at https://www.hl7.org/fhir/http.html and https://www.hl7.org/fhir/bundle.html.

In the table, each row corresponds to one resource that is created. To add the information for the `request` element of the bundle, this table has to be augmented with two columns named `request.method` and `request.url`, which contain the respective HTTP verb and URL for the resource. If these columns are not added to the table, `fhir_build_bundle()` still builds bundles from it, but those bundles will not be POSTable to a server. See examples.

## Value

A fhir_bundle_xml object.

## See Also

`fhir_crack()`,`fhir_cast()`, `fhir_build_resource()`, `fhir_post()`

## Examples

```
#unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

#crack fhir resources
table_desc_pat <- fhir_table_description(
    resource = "Patient",
    brackets = c("[", "]"),
    sep      = " ",
    format   = "wide"
)

df <- fhir_crack(bundles = bundles, design = table_desc_pat)

#add request info to table
request <- data.frame(
    request.method = c("POST", "PUT"),
    request.url    = c("Patient", "Patient/id3")
)

request_df <- cbind(df, request)

#build bundle
bundle <- fhir_build_bundle(table         = request_df,
                            brackets      = table_desc_pat@brackets,
                            resource_type = "Patient",
                            bundle_type   = "transaction")

#print to console
cat(toString(bundle))
```

---

`fhir_build_resource` *Build a single FHIR resource*

---

## Description

This function takes a single row from a wide table as produced by `fhir_crack()` and builds a fhir_resource_xml object from it. The column names of the table must represent the XPath expression of the respective element with indices for repeating items. A table like this is produced when FHIR resources have been cracked with `fhir_crack()` without assigning explicit column names in the fhir_design/fhir_table_description and with `format` set to `"wide"`.

## Usage

```
fhir_build_resource(row, brackets, resource_type)
```

### Arguments

| | |
|---|---|
| row | Single row from a wide table as produced by fhir_crack() with format="wide" |
| brackets | A character vector of length one. The brackets used for cracking. |
| resource_type | A character vector of length one or fhir_resource_type object indicating which resource type the table is build from. |

### Value

A fhir_resource_xml object.

### See Also

fhir_cast(), fhir_crack(), fhir_build_bundle(), fhir_post(), fhir_put()

### Examples

```
#unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

#crack fhir resources
Pat <- fhir_table_description(
    resource = "Patient",
    brackets = c("[", "]"),
    sep      = " ",
    format   = "wide"
)

df <- fhir_crack(bundles = bundles, design = Pat)


#build resource
resource <- fhir_build_resource(
                row           = df[1,],
                brackets      = c('[', ']'),
                resource_type = "Patient"
            )

#print to console
resource
```

---

## fhir_bundle-class *An S4 class to represent FHIR bundles*

---

### Description

An S4 class to represent FHIR bundles

`fhir_bundle_list` *Create fhir_bundle_list object*

## Description

A fhir_bundle_list is a list of fhir_bundle_xml or fhir_bundle_serialized objects. It is usually returned by a call to `fhir_search()`.

## Usage

```
fhir_bundle_list(bundles)
```

## Arguments

| | |
|---|---|
| bundles | A list of xml_nodes/fhir_bundle_xml objects or of raw/fhir_bundle_serialized objects |

## Details

The only scenario where one would use this constructor function is when several fhir_bundle or fhir_bundle_list objects should be merged into one big fhir_bundle_list before cracking (see examples).

## Examples

```
#unserialize example bundles
bundles1 <- fhir_unserialize(example_bundles1)
bundles2 <- fhir_unserialize(example_bundles2)

#bind them together in one fhir_bundle_list
bound_bundles <- fhir_bundle_list(c(bundles1, bundles2))
class(bound_bundles)

#bound list contains bundles from both original lists
length(bundles1)
length(bundles2)
length(bound_bundles)



#Create fhir_bundle list from xml objects
b1 <- xml2::read_xml("<Bundle><Resource><item value='1'/></Resource></Bundle>")
b2 <- xml2::read_xml("<Bundle><Resource><item value='2'/></Resource></Bundle>")

fhir_bundle_list(bundles = list(b1, b2))
fhir_bundle_list(bundles = list(fhir_bundle_xml(b1), fhir_bundle_xml(b2)))

r1 <- xml2::xml_serialize(object = b1, connection= NULL)
```

```
r2 <- xml2::xml_serialize(object = b2, connection= NULL)

fhir_bundle_list(bundles = list(r1, r2))
```

---

`fhir_bundle_list-class` *S4 class to represent a list of FHIR bundles*

---

## Description

A fhir_bundle_list is a list of fhir_bundle_xml or fhir_bundle_serialized objects. It should not be created by the user but returned by a call to `fhir_search()`.

---

`fhir_bundle_serialized-class` *An S4 class to represent a FHIR bundle in serialized form*

---

## Description

A `fhir_bundle_serialized` is a `fhir_bundle_xml` that has been serialized using `fhir_serialize()`. In this form, the bundle cannot be used in any meaningful way, but it can be saved and loaded as an `.RData` or `.rds` object without breaking the external pointers in the xml. See `?fhir_serialize` and `?fhir_unserialize`.

---

`fhir_bundle_xml` *Create fhir_bundle_xml object*

---

## Description

This should only be used if you want to create small examples. Usually, a `fhir_bundle_xml` will be returned by `fhir_search()`.

## Usage

```
fhir_bundle_xml(bundle)
```

## Arguments

| | |
|---|---|
| `bundle` | A xml-object representing a FHIR bundle |

## Examples

```
fhir_bundle_xml(bundle = xml2::xml_unserialize(patient_bundles[[1]]))
```

## fhir_bundle_xml-class *An S4 class to represent a FHIR bundle in xml form*

### Description

A fhir_bundle_xml is an xml representation of a FHIR bundle (https://www.hl7.org/fhir/bundle.html). It is usually found inside a fhir_bundle_list which is returned by a call to fhir_search().

### Slots

next_link A fhir_url pointing to the next bundle on the server.

self_link A fhir_url pointing to this bundle on the server.

## fhir_canonical_design *Retrieve design of last call to fhir_crack*

### Description

Returns the fhir_design of the last call to fhir_crack().

### Usage

```
fhir_canonical_design()
```

### See Also

fhir_design(), fhir_table_description()

### Examples

```
#load example bundles
bundles <- fhir_unserialize(bundles = patient_bundles)

#define design
patients <- fhir_table_description(resource = 'Patient')

design <- fhir_design(patients)

result <- fhir_crack(bundles = bundles, design = design)

fhir_canonical_design()
```

`fhir_capability_statement` *Get capability statement*

## Description

Get the capability statement of a FHIR server.

This function downloads a capability statement and creates three data.frames from it:

- `Meta` contains general information on the server
- `Rest` contains information on the Rest operations the server supports
- `Resources` contains information on the supported resource types

When there is more than one piece of information regarding a variable in these data.frames, they are divided by the string specified in `sep`. If `brackets` is not NULL, those entries will also be assigned indices so you can melt them using `fhir_melt()`.

## Usage

```
fhir_capability_statement(
  url = "https://hapi.fhir.org/baseR4",
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  brackets = NULL,
  sep = " ::: ",
  log_errors = NULL,
  verbose = deprecated()
)
```

## Arguments

| | |
|---|---|
| `url` | The base URL of the FHIR server. |
| `username` | A character vector of length one containing the username for basic authentication. Defaults to NULL, meaning no authentication. |
| `password` | A character vector of length one containing the password for basic authentication. Defaults to NULL, meaning no authentication. |
| `token` | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See `fhir_authenticate()` for how to create this. |
| `add_headers` | A named character vector of custom headers to add to the HTTP request, e.g. `c(myHeader = "somevalue")` or `c(firstHeader = "value1", secondHeader = "value2")`. |

| | |
|---|---|
| brackets | A character vector of length two defining the brackets surrounding indices for multiple entries, e.g. c( "[", "]"). Defaults to NULL. If NULL, no indices will be added to multiple entries. |
| sep | A character vector of length one to separate pasted multiple entries |
| log_errors | Either NULL or a character vector of length one indicating the name of a file in which to save the http errors. NULL means no error logging. When a file name is provided, the errors are saved in the specified file. Defaults to NULL |
| verbose | **[Deprecated]** |

## Value

A list of data frames containing the information from the statement

## Examples

```
## Not run:
#without indices
cap <- fhir_capability_statement(url = "https://server.fire.ly")

#with indices
cap <- fhir_capability_statement(url = "https://server.fire.ly",
                                 brackets = c("[","]"),
                                 sep = " || ")

#melt searchInclude variable
resources <- fhir_melt(cap$Resources,
                       columns = "searchInclude",
                       brackets = c("[", "]"),
                       sep = " || ",
                       all_columns = FALSE)

#remove indices
resources <- fhir_rm_indices(resources, brackets = c("[", "]"))

head(resources)

## End(Not run)
```

---

# fhir_cast

*Cast table with multiple entries This function divides multiple entries in a compact indexed table as produced by* fhir_crack() *into separate columns.*

---

## Description

This function turns a table from compact format into wide format. Every column containing multiple entries will be turned into multiple columns. The number of columns created from a single column in the original table is determined by the maximum number of multiple entries occurring in this column. Rows with less than the maximally occurring number of entries will be filled with NA values.

## Usage

```
fhir_cast(indexed_df, brackets, sep, verbose = 1)
```

## Arguments

| | |
|---|---|
| indexed_df | A compact data.frame/data.table with indexed multiple entries. Column names should reflect the XPath expression of the respective element. |
| brackets | A character vector of length two, defining the brackets used for the indices. |
| sep | A character vector of length one defining the separator that was used when pasting together multiple entries in fhir_crack(). |
| verbose | An integer vector of length one. If 0, nothing is printed, if 1, only general progress is printed, if > 1, progress for each variable is printed. Defaults to 1. |

## Details

For fhir_cast() to work properly, column names of the input data must reflect the Xpath to the corresponding resource element with . as a separator, e.g. code.coding.system. These names are produced automatically by fhir_crack() when the names are not explicitly set in the cols element of the fhir_table_description()/fhir_design().

In the names of the newly created columns the indices will be added in front of the column names, similar to the result of fhir_crack() with format="wide". See examples and the corresponding package vignette for a more detailed description.

## See Also

fhir_crack(), fhir_melt(), fhir_build_bundle()

## Examples

```
#unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

#crack fhir resources
table_desc <- fhir_table_description(
    resource = "Patient",
    brackets = c('[', ']'),
    sep      = " ",
    keep_attr=TRUE
)
df <- fhir_crack(bundles = bundles, design = table_desc)

#original df
df

#cast
fhir_cast(df, brackets=c('[', ']'), sep = ' ', verbose = 0)
```

fhir_columns *Create fhir_columns object*

## Description

An object of class fhir_columns is part of a fhir_table_description in a fhir_design and holds information on the elements that should be extracted from the FHIR resources, as well as the column names of the resulting data.frame. The elements to be extracted are indicated by XPath xpaths. If no column names are provided, they are generated automatically and reflect the elements position in the resource.

## Usage

```
fhir_columns(xpaths, colnames)

## S4 method for signature 'missing,missing'
fhir_columns()

## S4 method for signature '`NULL`,missing'
fhir_columns(xpaths)

## S4 method for signature 'character,character'
fhir_columns(xpaths, colnames)

## S4 method for signature 'character,missing'
fhir_columns(xpaths)

## S4 method for signature 'list,missing'
fhir_columns(xpaths)
```

## Arguments

| | |
|---|---|
| xpaths | A (named) character vector or (named) list containing xpath xpaths, or a fhir_xpath_expression object. |
| colnames | The names of the columns to create. If no colnames are provided and the list or vector in xpaths has names, those names are taken as the colnames. If no colnames are provided and xpaths is unnamed too, the colnames are generated automatically from the xpath xpaths. See examples. |

## Examples

```
#provide colnames explicitly
fhir_columns(xpaths = c("name/given", "code/coding/code"),
             colnames = c("name", "code"))

#colnames are taken from xpaths argument
fhir_columns(xpaths = c(name = "name/given", code = "code/coding/code"))
```

```
#colnames are taken from xpaths argument
fhir_columns(xpaths = list(name = "name/given", code = "code/coding/code"))

#colnames are generated automatically
fhir_columns(xpaths = c("name/given", "code/coding/code"))
```

---

`fhir_columns-class` *A S4 class to represent columns in a fhir_table_description*

---

## Description

An object of class `fhir_columns` is part of a fhir_table_description in a fhir_design and holds information on the elements that should be extracted from the FHIR resources, as well as the column names of the resulting data.frame. The elements to be extracted are indicated by XPath xpaths.

## Slots

`names` The column names

---

`fhir_common_columns` *Find common columns*

---

## Description

This is a convenience function to find all column names in a data frame starting with the same string that can then be used for `fhir_melt()`.

## Usage

```
fhir_common_columns(data_frame, column_names_prefix)
```

## Arguments

| | |
|---|---|
| `data_frame` | A data.frame/data.table with automatically named columns as produced by `fhir_crack()`. |
| `column_names_prefix` | A string containing the common prefix of the desired columns. |

## Details

It is intended for use on data frames with column names that have been automatically produced by `fhir_design()`/`fhir_crack()` and follow the form `level1.level2.level3` such as `name.given` or `code.coding.system`. Note that this function will only work on column names following exactly this scheme.

The resulting character vector can be used for melting all columns belonging to the same attribute in an indexed data frame, see `?fhir_melt`.

## Value

A character vector with the names of all columns matching `column_names_prefix`.

## See Also

`fhir_melt()`, `fhir_rm_indices()`

## Examples

```
#unserialize example bundles
bundles <- fhir_unserialize(bundles = medication_bundles)

#crack Patient Resources
pats <- fhir_table_description(resource = "Patient")

df <- fhir_crack(bundles = bundles, design = pats)

#look at automatically generated names
names(df)

#extract all column names beginning with the string "name"
fhir_common_columns(data_frame = df, column_names_prefix = "name")
```

---

# fhir_count_resource *Get Resources' Counts*

---

## Description

Downloads a count of resources matching the resource type and search parameters specified in `resource` and `parameters`. This function makes use of the `_summary=count` parameter of FHIR search and is therefore able to count resources on the server without actually downloading them.

## Usage

```
fhir_count_resource(
  base_url,
  resource,
  parameters = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL
)
```

## Arguments

| | |
|---|---|
| base_url | A character vector of length one specifying the base URL of the FHIR server, e.g. "http://hapi.fhir.org/baseR4". |
| resource | A character vector of length one or fhir_resource_type object with the resource type to be searched, e.g. "Patient". |
| parameters | Optional. Either a length 1 character vector containing properly formatted FHIR search parameters, e.g. "gender=male&_summary=count" or a named list or named character vector e.g. list(gender="male", "_summary"="count") or c(gender="male", "_summary"="count"). Note that parameter names beginning with _ have to be put in quotation marks! |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. c(myHeader = "somevalue") or c(firstHeader = "value1", secondHeader = "value2"). |

## Details

For more information on authentication options, please see the help page of fhir_search()

## Value

An integer of length 1 containing the number of resources matching the type and search parameters specified in resource and parameters.

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({

#number of female Patient resources on the server
fhir_count_resource(
  base_url = 'https://vonk.fire.ly/R4',
  resource = "Patient",
  parameters = c(gender = "female"))
}, silent = TRUE)
```

`fhir_crack` *Flatten list of FHIR bundles*

## Description

Converts a fhir_bundle_list (the result of `fhir_search()`) to a data.frame/data.table or list of df/dt, if more than one resource type is extracted at once.

There are two main output formats for the table: compact and wide. They differ regarding their handling of multiple occurrences of the same FHIR element (e.g. `Patient.adress`). In the compact format multiple occurrences are pasted together into one cell/column, in the wide format multiple occurrences are distributed over several (indexed) columns. If none of the resources contains any multiple values on the extracted elements, the two formats will result in the same structure.

To increase speed with larger amounts of data the cracking process can be parallelised over a number of cores defined in the `ncores` argument.

## Usage

```
fhir_crack(
  bundles,
  design,
  sep = NULL,
  brackets = NULL,
  rm_empty_cols = NULL,
  verbose = 2,
  data.table = FALSE,
  format = NULL,
  keep_attr = NULL,
  ncores = 1
)

## S4 method for signature 'ANY,fhir_table_description'
fhir_crack(
  bundles,
  design,
  sep = NULL,
  brackets = NULL,
  rm_empty_cols = NULL,
  verbose = 2,
  data.table = FALSE,
  format = NULL,
  keep_attr = NULL,
  ncores = 1
)

## S4 method for signature 'ANY,fhir_design'
fhir_crack(
```

```
  bundles,
  design,
  sep = NULL,
  brackets = NULL,
  rm_empty_cols = NULL,
  verbose = 2,
  data.table = FALSE,
  format = NULL,
  keep_attr = NULL,
  ncores = 1
)
```

## Arguments

| | |
|---|---|
| bundles | A FHIR search result as returned by fhir_search(). |
| design | A fhir_design or fhir_table_description object. See fhir_design()/fhir_table_description() and the corresponding vignette (vignette("flattenResources", package ="fhircrackr")) for a more detailed explanation and comprehensive examples of both. |
| sep | Optional. A character of length one containing the separator string used for separating multiple entries in cells when format = "compact". Will overwrite the sep defined in design. If sep = NULL, it is looked up in design, where the default is ":::". |
| brackets | Optional. A character of length one or two used for the indices of multiple entries, which will overwrite the brackets defined in design. If brackets = NULL, it is looked up in design, where the default is character(0),i.e. no indices are added to multiple entries. Empty strings ("") are not allowed. |
| rm_empty_cols | Optional. Remove empty columns? Logical scalar which will overwrite the rm_empty_cols defined in design. If rm_empty_cols = NULL, it is looked up in design, where the default is FALSE. |
| verbose | An integer vector of length one. If 0, nothing is printed, if 1, only finishing message is printed, if > 1, extraction progress will be printed. Defaults to 2. |
| data.table | A logical vector of length one. If it is set to TRUE the fhir_crack-function returns a data.table, otherwise a data.frame. Defaults to FALSE. |
| format | Optional. A character of length one indicating whether the resulting table should be cracked to a wide or compact format. Will overwrite the format defined in design which defaults to compact. wide means multiple entries will be distributed over several columns with indexed names. compact means multiple entries will be pasted into one cell/column separated by sep. |
| keep_attr | Optional. A logical of length one indicating whether the attribute name of the respective element (@value in most cases) should be attached to the name of the variable in the resulting table. Will overwrite keep_attr in design which defaults to FALSE. |
| ncores | Either NULL (no parallelisation) or an integer of length 1 containing the number of cpu cores that should be used for parallelised cracking. Parallelisation currently only works on linux systems. Defaults to NULL. |

## Value

If a fhir_design was used, the result is a list of data.frames, i.e. a fhir_df_list object, or a list of data.tables, i.e. a fhir_dt_list object. If a fhir_table_description was used, the result is a single data.frame/data.table.

## See Also

- Downloading bundles from a FHIR server: `fhir_search()`
- Creating designs/table_descriptions: `fhir_table_description()` and `fhir_design()`
- Dealing with multiple entries: `fhir_melt()`, `fhir_cast()`, `fhir_rm_indices()`

## Examples

```
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)


###Example 1###
#Extract just one resource type

#define attributes to extract
med_desc <- fhir_table_description(
   resource = "MedicationStatement",
   cols     = c(
    id              = "id",
    status          = "status",
    system          = "medicationCodeableConcept/coding/system",
    code            = "medicationCodeableConcept/coding/code",
    display         = "medicationCodeableConcept/coding/display"
  )
)

med_df <- fhir_crack(bundles = bundles, design = med_desc)

head(med_df) #data.frame


###Example 2###
#extract two resource types at once

pat_desc <- fhir_table_description(
   resource = "Patient"
)

design <- fhir_design(med_desc, pat_desc)

df_list <- fhir_crack(bundles = bundles, design = design)

#list of data.frames/fhir_df_list
head(df_list$med_desc)
head(df_list$pat_desc)
```

```
#The design that was used can be extracted from a fhir_df_list
fhir_design(df_list)


###Example 3###
#Filter values before extracting

#unserialize example bundle
b <- fhir_unserialize(bundles = example_bundles5)

#only extract codings with loinc system
table_desc <- fhir_table_description(
                    resource = "Observation",
                    cols = c(
                      id = "id",
                  loinc = "code/coding[system[@value='http://loinc.org']]/code",
                  display = "code/coding[system[@value='http://loinc.org']]/display"
                  )
)

table <- fhir_crack(bundles = b,
    design = table_desc)

table
```

---

`fhir_current_request` *Return FHIR search request used in last call to* `fhir_search()` *or* `fhir_url()`

---

## Description

Return FHIR search request used in last call to `fhir_search()` or `fhir_url()`

## Usage

```
fhir_current_request()
```

## Value

An object of class `fhir_url()`

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({
```

```
request <- fhir_url(url = "https://server.fire.ly", resource = "Patient")
fhir_current_request()

fhir_search("https://server.fire.ly/Medication", max_bundles = 1)
fhir_current_request()


},silent = TRUE)
```

---

`fhir_design` *Create a fhir_design object*

---

## Description

A `fhir_design` is a named list of `fhir_table_description` objects (See `fhir_table_description()`) and should be created using the function described here. The design is used in `fhir_crack()` to tell the function how to flatten each resource type.

## Usage

```
fhir_design(...)

## S4 method for signature 'fhir_table_description'
fhir_design(...)

## S4 method for signature 'list'
fhir_design(...)

## S4 method for signature 'fhir_table_list'
fhir_design(...)
```

## Arguments

| | |
|---|---|
| `...` | One or more `fhir_table_description` objects or a named list containing `fhir_table_description` objects, or an object of class fhir_df_list/fhir_dt_list. See `fhir_table_description()`. |

## Details

A `fhir_design` looks for example like this:

```
A fhir_design with 2 table_descriptions:
A fhir_table_description with the following elements:
```

```
fhir_resource_type: Patient

fhir_columns:
------------ -----------------
column name | xpath expression
------------ -----------------
id          | id
name        | name/family
gender      | gender
------------ -----------------

sep:           '||'
brackets:      '[', ']'
rm_empty_cols: FALSE
format:        'compact'
keep_attr:     TRUE

A fhir_table_description with the following elements:

fhir_resource_type: MedicationAdministration

fhir_columns:
An empty fhir_columns object

sep:           ':::'
brackets:      no brackets
rm_empty_cols: FALSE
format:        'wide'
keep_attr:     TRUE
```

The names of the table_descriptions are taken from the names of the arguments. If the table_descriptions are created within the call to `fhir_design` and therefore have no names, the names will be created from the respective resource type. See examples.

For backwards compatibility it is for the moment also possible to build it from an old-style design as used in `fhircrackr (< 1.0.0)`. See examples.

If this function is given an object of class fhir_df_list or fhir_dt_list, it will extract the design that was used to create the respective list.

## See Also

`fhir_table_description()`, `fhir_crack()`

## Examples

```
####Example 1####

###create fhir_table_descriptions first
#see ?fhir_table_description for explanation
```

```
pat <- fhir_table_description(
    resource      = "Patient",
    cols          = c(
        id            = "id",
        name          = "name/family",
        gender        = "gender"
    ),
    sep           = "||",
    brackets      = c("[", "]"),
    rm_empty_cols = FALSE

)

meds <- fhir_table_description(resource = "MedicationAdministration")

###create design
#First option: Explicitly define names

design1 <- fhir_design(Pats = pat, Medics = meds)
print(design1)


#Second option: Names are taken from the object names

design2 <- fhir_design(pat, meds)
print(design2)


#Third option: Create table_description within fhir_design

design3 <- fhir_design(fhir_table_description(resource = "MedicationAdministration"))
print(design3)


#Fourth option: Names are taken from named list

design3 <- fhir_design(list(Patients = pat, Medications = meds))
print(design3)


###Example 2###
###Extract design from fhir_df_list/fhir_dt_list

#unserialize and crack example bundles
med_bundles <- fhir_unserialize(bundles = medication_bundles)
dfs <- fhir_crack(bundles = med_bundles, design = design1)

#extract design

fhir_design(dfs)
```

## fhir_design-class *A S4 class containing a design for* fhir_crack()

### Description

A fhir_design is a named list of fhir_table_description objects. Each table_description contains information on how to flatten one resource type which will result in one data.frame. The fhir_design is passed to the function fhir_crack() along with a list of bundles containing FHIR resources.

### Slots

`.Data` The list of `fhir_table_description` objects.

`names` The names of the table_descriptions. Those will also be the names of the resulting data.frames.

### See Also

fhir_table_description(), fhir_crack()

## fhir_df_list-class *List of data.frames as returned by* fhir_crack()

### Description

Objects of this class are returned by fhir_crack() when `data.table=FALSE` (the default). They behave like an ordinary named list of data.frames but have some additional information in the slot `design`.

### Slots

`names` Character vector containing the names of the data.frames.

`design` An object of class fhir_design that was used to create the df_list.

## fhir_dt_list-class *List of data.tables as returned by* fhir_crack()

### Description

Objects of this class are returned by fhir_crack() when `data.table=TRUE`. They behave like an ordinary named list of data.tables but have some additional information in the slot `design`.

### Slots

`names` A character vector containing the names of the data.tables.

`design` An object of class fhir_design that was used to create the dt_list.

fhir_get_resources_by_ids *Get Resources by their IDs*

## Description

Downloads FHIR resources represented by a vector of resource IDs.

## Usage

```
fhir_get_resources_by_ids(
  base_url,
  resource,
  ids,
  id_param = "_id",
  parameters = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  verbose = 0
)
```

## Arguments

| | |
|---|---|
| base_url | A character vector of length one specifying the base URL of the FHIR server, e.g. "http://hapi.fhir.org/baseR4". |
| resource | A character vector of length one or fhir_resource_type object with the resource type to be searched, e.g. "Patient". |
| ids | A character vector containing the IDs of the resources that should be downloaded. In the default setting these should be resource (aka logical) IDs. |
| id_param | A character vector of length one containing the FHIR Search parameter belonging to the ids in ids. Defaults to "_id" meaning ids is interpreted as containing resource (aka logical) ids. Could be changed to "identifier" if ids contains a vector of identifier values instead. |
| parameters | FHIR Search parameters to further restrict the set of resources that is returned, e.g. gender=male to only download the resources from the ids list that correspond to males. Can be either a length 1 character containing properly formatted FHIR search parameters, e.g. "gender=male" or a named list or named character vector e.g. list(gender="male")or c(gender="male"). Defaults to NULL meaning no restriction on the IDs provided in ids. |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |

| | |
|---|---|
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. c(myHeader = "somevalue") or c(firstHeader = "value1", secondHeader = "value2"). |
| verbose | An integer vector of length 1 containing the level of verbosity. Defaults to 0. |

## Details

This function takes a character vector ids containing logical Ids of resources of a given type (specified in resource) on a FHIR server (specified in base_url) and downloads the corresponding resources from the server. The function will attempt to download the resources using a FHIR search request via POST where the IDs are part of the body. See fhir_search() for details. If this fails (e.g. because the server doesn't allow POST operations), the function falls back on a GET request. If the set of ids is too long to fit into one GET request (i.e. if the request gets longer than 2083 characters), it will be spread across several requests.

For more information on authentication options, please see the help page of fhir_search()

## Value

A fhir_bundle_list containing the downloaded resources.

## See Also

fhir_search(), fhir_get_resource_ids()

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({

#find IDs of Patient resources representing Homer Simpson
ids <- fhir_get_resource_ids(
  base_url   = 'https://hapi.fhir.org/baseR4',
  resource   = "Patient",
  parameters = "name=Homer&name=Simpson")

#Download all corresponding resources
bundles <- fhir_get_resources_by_ids(
  base_url = 'https://hapi.fhir.org/baseR4',
  resource = "Patient",
  ids      = ids)

#have a look at the resources
fhir_crack(
```

```
  bundles,
  fhir_table_description(
    resource = "Patient",
    cols     = list(
      ID      = "id",
      given   = "name/given",
      family  = "name/family")))

}, silent = TRUE)
```

---

`fhir_get_resource_ids` *Get Resources' IDs*

---

## Description

Download the resource (aka logical) IDs of all resources matching the FHIR search request build from the resource type and search parameters specified in `resource` and `parameters`. This function does not download the entire resources, but only extracts their IDs using the `_elements` parameter of FHIR Search.

## Usage

```
fhir_get_resource_ids(
  base_url,
  resource,
  parameters = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  verbose = 0
)
```

## Arguments

| | |
|---|---|
| `base_url` | A character vector of length one specifying the base URL of the FHIR server, e.g. `"http://hapi.fhir.org/baseR4"`. |
| `resource` | A character vector of length one or fhir_resource_type object with the resource type to be searched, e.g. `"Patient"`. |
| `parameters` | Optional. Either a length 1 character vector containing properly formatted FHIR search parameters, e.g. `"gender=male&_summary=count"` or a named list or named character vector e.g. `list(gender="male", "_summary"="count")` or `c(gender="male", "_summary"="count")`. Note that parameter names beginning with _ have to be put in quotation marks! |
| `username` | A character vector of length one containing the username for basic authentication. |

| | |
|---|---|
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. c(myHeader = "somevalue") or c(firstHeader = "value1", secondHeader = "value2"). |
| verbose | An integer of length 1 containing the level of verbosity. Defaults to 0. |

## Details

For more information on authentication options, please see the help page of fhir_search()

## Value

A character vector containing the resource (aka logical) IDs of all requested resources.

## See Also

fhir_search(), fhir_get_resources_by_ids()

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({

fhir_get_resource_ids(
  base_url   = 'https://vonk.fire.ly/R4',
  resource   = "Patient",
  parameters = "gender=female", verbose=1)

  }, silent = TRUE)
```

---

# fhir_load *Load bundles from xml-files*

---

## Description

Reads all bundles stored as xml files from a directory.

### Usage

```
fhir_load(directory, indices = NULL, pattern = "^[0-9]+\\.xml$")
```

### Arguments

| | |
|---|---|
| directory | A character vector of length one containing the path to the folder were the files are stored. |
| indices | A numeric vector of integers indicating which bundles from the specified directory should be loaded. Defaults to NULL meaning all bundles from the directory are loaded. |
| pattern | A character vector of length one with a regex expression defining the naming pattern of the xml files to be read. Defaults to the regex expression matching file names as produced by fhir_save(). |

### Value

A fhir_bundle_list.

### Examples

```
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)
length(bundles)

#save to temporary directory
dir <- tempdir()
fhir_save(bundles, directory = dir)

#load from temporary directory
loaded_bundles <- fhir_load(dir)
length(loaded_bundles)

#load only two, the second and the third bundle
loaded_bundles <- fhir_load(dir, indices = c(2,3))
length(loaded_bundles)
```

---

## fhir_load_design *Load design from xml*

---

### Description

Loads a fhir_design for use with fhir_crack() from an xml file into R.

### Usage

```
fhir_load_design(file)
```

## Arguments

| | |
|---|---|
| file | A string specifying the file from which to read. |

## Value

A fhir_design object. See ?fhir_design.

## See Also

fhir_design(), fhir_table_description(), fhir_save_design()

## Examples

```
table_desc1 <- fhir_table_description(
    resource = 'Patient',
    cols     = c(
        id     = 'id',
        name   = 'name/family',
        gender = 'gender'
    ),
    sep          = ':::',
    brackets     = c('[', ']'),
    rm_empty_cols = FALSE,
    format       = 'compact',
    keep_attr    = FALSE
)

table_desc2 <- fhir_table_description(
    resource = 'Observation',
    cols     = c(
        'code/coding/system',
        'code/coding/code'
    )
)

design <- fhir_design(
    Patients     = table_desc1,
    Observations = table_desc2
)
temp <- tempfile()

fhir_save_design(design = design, file = temp)

design <- fhir_load_design(file = temp)
```

`fhir_melt` *Melt multiple entries*

## Description

This function divides multiple entries in an indexed data frame as produced by `fhir_crack()` into separate rows.

## Usage

```
fhir_melt(
  indexed_data_frame,
  columns,
  brackets = c("<", ">"),
  sep = " ",
  id_name = "resource_identifier",
  all_columns = FALSE
)
```

## Arguments

| | |
|---|---|
| `indexed_data_frame` | A data.frame/data.table with indexed multiple entries. |
| `columns` | A character vector specifying the names of all columns that should be molten simultaneously. It is advisable to only melt columns simultaneously that belong to the same (repeating) attribute! |
| `brackets` | A character vector of length two, defining the brackets used for the indices. |
| `sep` | A character vector of length one defining the separator that was used when pasting together multiple entries in `fhir_crack()`. |
| `id_name` | A character vector of length one, the name of the column that will hold the identification of the origin of the new rows. |
| `all_columns` | Return all columns? Defaults to `FALSE`, meaning only those specified in `columns` are returned. |

## Details

Every row containing values that consist of multiple entries on the variables specified by the argument `columns` will be turned into multiple rows, one for each entry. Values on other variables will be repeated in all the new rows.

The new data.frame will contain only the molten variables (if `all_cloumns = FALSE`) or all variables (if `all_columns = TRUE`) as well as an additional variable `resource_identifier` that maps which rows came from the same origin. The name of this column can be changed in the argument `id_name`.

For a more detailed description on how to use this function please see the corresponding package vignette.

## Value

A data.frame/data.table where each entry from the variables in `columns` appears in a separate row.

## See Also

`fhir_common_columns()`, `fhir_rm_indices()`

## Examples

```
#unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

#crack fhir resources
table_desc <- fhir_table_description(
    resource = "Patient",
    brackets = c("[", "]"),
    sep = " "
)

df <- fhir_crack(bundles = bundles, design = table_desc)

#find all column names associated with attribute address
col_names <- fhir_common_columns(df, "address")

#original data frame
df

#only keep address columns
fhir_melt(
     indexed_data_frame = df,
     columns            = col_names,
     brackets           = c("[", "]"),
     sep = " "
 )

#keep all columns
fhir_melt(indexed_data_frame = df, columns = col_names,
          brackets = c("[","]"), sep = " ", all_columns = TRUE)
```

---

# `fhir_next_bundle_url` *Next Bundle's URL*

---

## Description

fhir_next_bundle_url() gives the link to the next available bundle, either of the bundle you provided in the argument `bundle` or of the last call to `fhir_search()`, if `bundle=NULL` (the default).

This function is useful when you don't have a lot of memory available or when a download of bundles was interrupted for some reason. In case of small memory, you can use `fhir_next_bundle_url` together with the `max_bundle` argument from `fhir_search()` to download bundles in smaller batches in a loop. See details in the example.

## Usage

```
fhir_next_bundle_url(bundle = NULL)
```

## Arguments

| | |
|---|---|
| bundle | The bundle from which you wish to extract the next link. If this is NULL (the default), the function will extract the next link from the last bundle that was downloaded in the most recent call to fhir_search(). |

## Value

A fhir_url object referencing next bundle available on the FHIR server. Empty fhir_url / character vector, if no further bundle is available.

## Examples

```
#' #the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({


# workflow for small memory environments, downloading small batches of bundles
# for really small memory environments consider also using the `_count` option in
# your FHIR search request.
# You can iteratively download, crack and save the bundles until all bundles are processed or the
# desired number of bundles is reached.
url <- fhir_url("https://server.fire.ly/Patient")
count <- 0
obs <- fhir_table_description(resource = "Patient")
design <- fhir_design(obs)
while(length(url)>0 && count < 5){
 bundles <- fhir_search(url, max_bundles = 2)
 tables <- fhir_crack(bundles, design)
  save(tables, file = paste0(tempdir(),"/table_", count, ".RData"))
  count <- count + 1
  url <- fhir_next_bundle_url()
}
#you can see the saved tables here:
dir(tempdir())


}, silent = TRUE)
```

fhir_post *POST to a FHIR server*

## Description

This function is a convenience wrapper around `httr::POST()`.

## Usage

```
fhir_post(
  url,
  body,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  verbose = 1,
  log_errors = NULL
)

## S4 method for signature 'ANY,fhir_resource'
fhir_post(
  url,
  body,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  verbose = 1,
  log_errors = NULL
)

## S4 method for signature 'ANY,fhir_bundle_xml'
fhir_post(
  url,
  body,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  verbose = 1,
  log_errors = NULL
)

## S4 method for signature 'ANY,fhir_body'
fhir_post(
  url,
```

```
  body,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  verbose = 1,
  log_errors = NULL
)
```

## Arguments

| | |
|---|---|
| url | An object of class fhir_url or a character vector of length one containing the url to POST to. |
| body | An object of class fhir_resource, fhir_bundle_xml or fhir_body. See details for how to generate them. |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See `fhir_authenticate()` for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. `c(myHeader = "somevalue")` or `c(firstHeader = "value1", secondHeader = "value2")`. |
| verbose | An integer vector of length one. If 0, nothing is printed, if > 0 success message is printed. Defaults to 1. |
| log_errors | Either `NULL` or a character vector of length one indicating the name of a file in which to save http errors. `NULL` means no error logging. When a file name is provided, the errors are saved in the specified file. Defaults to `NULL`. Regardless of the value of `log_errors` the most recent http error message within the current R session is saved internally and can be accessed with `fhir_recent_http_error()`. |

## Details

`fhir_post()` accepts four classes for the body:

1. A fhir_resource as created by `fhir_build_resource()`. This is used when just a single resource should be POSTed to the server. In this case `url` must contain the base url plus the resource type, e.g. http://hapi.fhir.org/baseR4/Patient.
2. A fhir_bundle_xml representing a transaction or batch bundle as created by `fhir_build_bundle()`.
3. A fhir_body as created by `fhir_body()`. This is the most flexible approach, because within the fhir_body object you can represent any kind of `content` as a string and set the `type` accordingly. See examples.

For examples of how to create the different body types see the respective help pages. For an example of the entire workflow around creating and POSTing resources, see the package vignette on recreating resources.

## Examples

```
## Not run:
### 1. POST transaction bundle
#unserialize example bundles
bundle <- fhir_unserialize(transaction_bundle_example)

#have a look at the bundle
cat(toString(bundle))

#post
fhir_post(url = "http://hapi.fhir.org/baseR4", body = bundle)


### 2. POST single resouce
#unserialize example resource
resource <- fhir_unserialize(example_resource1)

#have a look at the resource
resource

#post
url <- fhir_url(url = "http://hapi.fhir.org/baseR4", resource = "Patient")
fhir_post(url = url, body = resource)


### 3. POST arbitrary body
#define body
body <- fhir_body(content = "<Patient> <gender value='female'/> </Patient>", type = "xml")

#post
url <- fhir_url(url = "http://hapi.fhir.org/baseR4", resource = "Patient")
fhir_post(url = url, body = body)

## End(Not run)
```

---

`fhir_put` *PUT to a FHIR server*

---

## Description

This function is a convenience wrapper around `httr::PUT()`.

## Usage

```
fhir_put(
  url,
  body,
  username = NULL,
  password = NULL,
```

```
  token = NULL,
  add_headers = NULL,
  verbose = 1,
  log_errors = NULL
)
```

## Arguments

| | |
|---|---|
| url | An object of class fhir_url or a character vector of length one containing the url to PUT to. |
| body | An object of class fhir_resource or fhir_body. See details for how to generate them. |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See `fhir_authenticate()` for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. `c(myHeader = "somevalue")` or `c(firstHeader = "value1", secondHeader = "value2")`. |
| verbose | An integer vector of length one. If 0, nothing is printed, if > 0 success message is printed. Defaults to 1. |
| log_errors | Either `NULL` or a character vector of length one indicating the name of a file in which to save http errors. `NULL` means no error logging. When a file name is provided, the errors are saved in the specified file. Defaults to `NULL`. Regardless of the value of `log_errors` the most recent http error message within the current R session is saved internally and can be accessed with `fhir_recent_http_error()`. |

## Details

`fhir_put()` accepts two classes for the body:

1. A fhir_resource as created by `fhir_build_resource()`. This is used when just a single resource should be PUT to the server. In this case `url` must contain the base url plus the resource type and the resource id, e.g. http://hapi.fhir.org/baseR4/Patient/1a2b3c.
2. A fhir_body as created by `fhir_body()`. This is the most flexible approach, because within the fhir_body object you can represent any kind of `content` as a string and set the `type` accordingly. See examples.

For examples of how to create the different body types see the respective help pages. For an example of the entire workflow around creating and PUTing resources, see the package vignette on recreating resources.

## Examples

```
## Not run:
### 1. PUT fhir__resource object
#unserialize example resource
resource <- fhir_unserialize(example_resource2)

#have a look at the resource
resource

#put
fhir_put(url = "http://hapi.fhir.org/baseR4/Patient/1a2b3c", body = resource)

### 2. PUT fhir_body object
#define body
body <- fhir_body(content = "<Patient> <id value='x1y2'/> <gender value='female'/> </Patient>",
                  type = "xml")

#put
fhir_put(url = "http://hapi.fhir.org/baseR4/Patient/x1y2", body = body)

## End(Not run)
```

---

# fhir_recent_http_error

*Return most recent http error from* fhir_search()

---

## Description

Whenever a call to fhir_search() produces any http error, the error information is saved internally until the next http error occurs (or the R session ends). The error information can be accessed with fhir_recent_http_error. If you want to log that information outside of your R session, set the argument log_errors of fhir_search() .

## Usage

```
fhir_recent_http_error()
```

## Value

A string containing the error message

## See Also

fhir_search()

## Examples

```
## Not run:
fhir_search("https://server.fire.ly/Medicatio", max_bundles = 1)
cat(fhir_recent_http_error())

## End(Not run)
```

---

`fhir_request` *fhir_request*

---

## Description

A Wrapper for fhir_url

## Usage

```
fhir_request(url, resource = NULL, parameters = NULL, url_enc = TRUE)
```

## Arguments

| | |
|---|---|
| `url` | The same as for `fhir_url()`. |
| `resource` | The same as for `fhir_url()`. Defaults to NULL. |
| `parameters` | The same as for `fhir_url()`. Defaults to NULL. |
| `url_enc` | The same as for `fhir_url()`. Defaults to TRUE. |

## Value

The same as for `fhir_url()`.

## Examples

```
#provide full FHIR search request
fhir_request(url = "http://hapi.fhir.org/baseR4/Patient?gender=male&_summary=count")

#provide base url and resource type
fhir_request(
   url      = "http://hapi.fhir.org/baseR4",
   resource = "Patient"
 )

#parameters in one string
fhir_request(
   url        = "http://hapi.fhir.org/baseR4",
   resource   = "Patient",
   parameters = "gender=male&_summary=count"
 )
```

```
#parameters as a named character
fhir_request(
   url        = "http://hapi.fhir.org/baseR4",
   resource   = "Patient",
   parameters = c("gender" = "male", "_summary" = "count")
 )

#parameters as a named list
fhir_request(
   url        = "http://hapi.fhir.org/baseR4",
   resource   = "Patient",
   parameters = list("gender" = "male", "_summary" = "count")
 )
```

---

## fhir_resource-class *An S4 class to represent FHIR resources*

---

### Description

An S4 class to represent FHIR resources

---

## fhir_resource_serialized-class *An S4 class to represent a FHIR resource in serialized form*

---

### Description

A `fhir_resource_serialized` is a `fhir_resource_xml` that has been serialized using `fhir_serialize()`. In this form, the resource cannot be used in any meaningful way, but it can be saved and loaded as an `.RData` or `.rds` object without breaking the external pointers in the xml. See `?fhir_serialize` and `?fhir_unserialize`.

---

## fhir_resource_type *Create fhir_resource_type object*

---

### Description

This function creates an object of class fhir_resource_type. It checks the resource type against the list of resource types provided at https://hl7.org/FHIR/resourcelist.html, corrects wrong cases (which can be disabled with `fix_capitalization = FALSE`) and throws a warning if the resource cannot be found at hl7.org.

### Usage

```
fhir_resource_type(string, fix_capitalization = TRUE)
```

### Arguments

| | |
|---|---|
| string | A length one character vector containing the resource type. Will usually be one of the official FHIR resource types listed at https://hl7.org/FHIR/resourcelist.html |
| fix_capitalization | Correct wrong capitalization for known resource types? E.g. patients -> Patients or medicationstatement -> MedicationStatement. Defaults to TRUE. |

### Value

An fhir_resource_type object

### Examples

```
fhir_resource_type(string = "Patient")
fhir_resource_type(string = "medicationadministration")
```

---

## fhir_resource_type-class *A representation of a FHIR resource type*

---

### Description

An object of class fhir_resource_type is a string containing a FHIR resource type. It is part of a fhir_table_description which in turn is part of a fhir_design and used in fhir_crack().

---

## fhir_resource_xml *Create fhir_resource_xml object*

---

### Description

Create fhir_resource_xml object

### Usage

```
fhir_resource_xml(resource)

## S4 method for signature 'xml_document'
fhir_resource_xml(resource)

## S4 method for signature 'xml_node'
fhir_resource_xml(resource)

## S4 method for signature 'character'
fhir_resource_xml(resource)
```

### Arguments

| | |
|---|---|
| resource | A xml-object representing a FHIR resource |

### Examples

```
fhir_resource_xml(resource = xml2::read_xml("<Patient><id value = '1'/></Patient>"))
```

---

## fhir_resource_xml-class *An S4 class to represent a FHIR resource in xml form*

---

### Description

A fhir_resource_xml is an xml representation of a FHIR resource (https://www.hl7.org/fhir/resourcelist.html).

---

## fhir_rm_div *Remove html elements*

---

### Description

This function is a convenience wrapper for fhir_rm_tag() that removes all   elements from an xml. div tags in FHIR resources contain html code, which is often server generated and in most cases neither relevant nor usable for data analysis.

### Usage

```
fhir_rm_div(x)
```

### Arguments

| | |
|---|---|
| x | A fhir_bundle_xml or fhir_bundle_list object or a character vector containing xml objects. |

### Value

An object of the same class as x where all tags matching the tag argument are removed.

### See Also

fhir_rm_tag()

## Examples

```
#Example 1: Remove div tags from xmls in a character vector
string <- c("Hallo<div>Please<p>Remove Me</p></div> World!",
            "A<div><div><p>B</p></div>C</div>D")

fhir_rm_div(x = string)



#Example 2: Remove div tags in a single fhir bundle
bundle <- fhir_unserialize(patient_bundles)[[1]]

#example bundle contains html parts in div tags:
cat(toString(bundle))

#remove html parts
bundle_cleaned <- fhir_rm_div(x = bundle)

#have a look at the result
cat(toString(bundle_cleaned))




#Example 3: Remove div tags in a list of fhir bundles
bundle_list <- fhir_unserialize(patient_bundles)


#remove html parts
bundle_list_cleaned <- fhir_rm_div(x = bundle_list)

#check out how much the size of the bundle list is reduced by removing html
size_with_html <- sum(sapply(bundle_list, function(x)object.size(toString(x))))
size_without_html <- sum(sapply(bundle_list_cleaned, function(x)object.size(toString(x))))

size_without_html/size_with_html
```

---

`fhir_rm_indices` *Remove indices from data.frame/data.table*

---

## Description

Removes the indices in front of multiple entries as produced by `fhir_crack()` when brackets are provided in the design.

## Usage

```
fhir_rm_indices(
  indexed_data_frame,
  brackets = c("<", ">"),
  columns = names(indexed_data_frame)
)
```

## Arguments

| | |
|---|---|
| indexed_data_frame | A data frame with indices for multiple entries as produced by fhir_crack() |
| brackets | A character vector of length two defining the brackets that were used in fhir_crack() |
| columns | A character vector of column names, indicating from which columns indices should be removed. Defaults to all columns. |

## Value

A data frame without indices.

## See Also

fhir_melt()

## Examples

```
#unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

patients <- fhir_table_description(resource = "Patient")

df <- fhir_crack(bundles = bundles,
                 design = patients,
                 brackets = c("[", "]"))

df_indices_removed <- fhir_rm_indices(indexed_data_frame = df, brackets=c("[", "]"))
```

---

# fhir_rm_tag *Remove a certain xml tag*

---

## Description

Removes a given xml tag from xml objects represented in a fhir_bundle_xml, fhir_bundle_list or character vector.

## Usage

```
fhir_rm_tag(x, tag)

## S4 method for signature 'character'
fhir_rm_tag(x, tag)

## S4 method for signature 'fhir_bundle_xml'
fhir_rm_tag(x, tag)

## S4 method for signature 'fhir_bundle_list'
fhir_rm_tag(x, tag)
```

## Arguments

| | |
|---|---|
| x | A fhir_bundle_xml or fhir_bundle_list object or a character vector containing xml objects. |
| tag | A character vector of length 1 containing the tag that should be removed, e.g. "div". |

## Details

In the example Hello Please Remove Me World! one could for example remove the tag p, resulting in Hello Please World!or remove the div tag resulting in Hello World!.

## Value

An object of the same class as x where all tags matching the tag argument are removed.

## See Also

fhir_rm_div()

## Examples

```
#Example 1: Remove tag from xmls in a character vector
string <- c("Hello<div>Please<p>Remove Me</p></div> World!",
            "A<div><div><p>B</p></div>C</div>D")

fhir_rm_tag(x = string, tag = "p")



#Example 2: Remove div tags in a single fhir bundle
bundle <- fhir_unserialize(patient_bundles)[[1]]

#example bundle contains html parts in div tags:
cat(toString(bundle))
```

```
#remove html parts
bundle_cleaned <- fhir_rm_tag(x = bundle, tag = "div")

#have a look at the result
cat(toString(bundle_cleaned))




#Example 3: Remove div tags in a list of fhir bundles
bundle_list <- fhir_unserialize(patient_bundles)


#remove html parts
bundle_list_cleaned <- fhir_rm_tag(x = bundle_list, tag = "div")

#check out how much the size of the bundle list is reduced by removing html
size_with_html <- sum(sapply(bundle_list, function(x)object.size(toString(x))))
size_without_html <- sum(sapply(bundle_list_cleaned, function(x)object.size(toString(x))))

size_without_html/size_with_html
```

---

`fhir_sample_resources` *Randomly sample resources from a FHIR server*

---

## Description

Downloads a random sample of resources of a given resource type from a FHIR server. The resources can be further constrained using FHIR search parameters.

## Usage

```
fhir_sample_resources(
  base_url,
  resource,
  parameters = NULL,
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  sample_size = 20,
  seed = 1,
  verbose = 1
)
```

## Arguments

| | |
|---|---|
| base_url | A character vector of length one specifying the base URL of the FHIR server, e.g. "http://hapi.fhir.org/baseR4". |
| resource | A character vector of length one or fhir_resource_type object with the resource type to be downloaded, e.g. "Patient". |
| parameters | Optional. Either a length 1 character vector containing properly formatted FHIR search parameters, e.g. "gender=male&_summary=count" or a named list or named character vector e.g. list(gender="male", "_summary"="count") or c(gender="male", "_summary"="count"). Note that parameter names beginning with _ have to be put in quotation marks! |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. c(myHeader = "somevalue") or c(firstHeader = "value1", secondHeader = "value2"). |
| sample_size | A integer of length 1 containing the number of resources to sample. |
| seed | A integer of length 1 containing the seed for the random generator. |
| verbose | An integer of length 1 containing the level of verbosity. Defaults to 1. |

## Details

This function performs three steps to draw a random sample of resources from a FHIR server:

1. Count how many resources matching the type resource and the search parameters in parameters are found on the server. This is done to assert that the desired sample_size is bigger than the number of resources it is drawn from. This step can also be performed individually using fhir_count_resource().
2. Extract the resource (aka logical) IDs of all requested resources (without downloading the resources completely). This step can be also be performed individually using fhir_get_resource_ids()
3. Draw a random sample of size sample_size from the vector of resource IDs and download the corresponding set of resources from the server. This can also be done individually using fhir_sample_resources_by_ids()

The actual download of the resources is done by fhir_get_resources_by_ids(). This function will attempt to download the resources using a FHIR search request via POST where the IDs are part of the body. See fhir_search() for details. If this fails (e.g. because the server doesn't allow POST operations), the function falls back on a GET request. If the set of IDs is too long to fit into one GET request (i.e. if the request gets longer than 2083 characters), it will be spread across several requests.

For more information on authentication options, please see the help page of fhir_search()

## Value

A fhir_bundle_list containing randomly sampled resources.

## See Also

fhir_search(), fhir_sample_resources_by_ids(), fhir_get_resources_by_ids(), fhir_count_resource()

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({

#how many resources are on the server?
count <- fhir_count_resource(
  base_url    = 'https://hapi.fhir.org/baseR4',
  resource    = "Patient",
  parameters  = "gender=female")

#randomly sample 30 of them
bundles <- fhir_sample_resources(
  base_url    = 'https://hapi.fhir.org/baseR4',
  resource    = "Patient",
  parameters  = "gender=female",
  sample_size = 30,
  seed        = 1)

bundles

}, silent = TRUE)
```

---

# fhir_sample_resources_by_ids

*Download a random sample of resources from a vector of resource IDs.*

---

## Description

Download a random sample of resources from a vector of resource IDs.

## Usage

```
fhir_sample_resources_by_ids(
  base_url,
  resource,
```

```
  ids,
  id_param = "_id",
  username = NULL,
  password = NULL,
  token = NULL,
  add_headers = NULL,
  sample_size = 20,
  seed = 1,
  verbose = 1
)
```

## Arguments

| | |
|---|---|
| base_url | A character vector of length one specifying the base URL of the FHIR server, e.g. "http://hapi.fhir.org/baseR4". |
| resource | A character vector of length one or fhir_resource_type object with the resource type to be downloaded e.g. "Patient". |
| ids | A character vector containing the IDs from which to sample. |
| id_param | A character vector of length one containing the FHIR Search parameter belonging to the ids in ids. Defaults to "_id" meaning ids is interpreted as containing resource (aka logical) ids. Could be changed to "identifier" if ids contains a vector of identifier values instead. |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. c(myHeader = "somevalue") or c(firstHeader = "value1", secondHeader = "value2"). |
| sample_size | A integer of length 1 containing the number of resources to sample. |
| seed | A integer of length 1 containing the seed for the random generator. |
| verbose | An integer of length 1 containing the level of verbosity. Defaults to 1. |

## Details

This function takes a character vector ids containing logical Ids of resources of a given type (specified in resource) on a FHIR server (specified in base_url) and downloads a random sample of size sample_size of the corresponding resources from the server.

Internally, the download of the resources is done by fhir_get_resources_by_ids(). This function will attempt to download the resources using a FHIR search request via POST where the IDs are part of the body. See fhir_search() for details. If this fails (e.g. because the server doesn't allow POST operations), the function falls back on a GET request. If the set of IDs is too long to fit

into one GET request (i.e. if the request gets longer than 2083 characters), it will be spread across several requests.

For more information on authentication options, please see the help page of 'fhir_search()

## Value

A list of bundles containing sampled resources.

## See Also

fhir_search(), fhir_sample_resources(), fhir_get_resources_by_ids(), fhir_count_resource()

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({

#find IDs of all resources representing Homer Simpson
ids <- fhir_get_resource_ids(
  base_url   = 'https://hapi.fhir.org/baseR4',
  resource   = "Patient",
  parameters = "name=Homer&name=Simpson")

#Sample 10 of them
bundles <- fhir_sample_resources_by_ids(
  base_url    = 'https://hapi.fhir.org/baseR4',
  resource    = "Patient",
  ids         = ids,
  sample_size = 10,
  seed        = 1)

#Have a look at the samples
fhir_crack(
  bundles,
  fhir_table_description(
    resource = "Patient",
    cols     = list(
      ID     = "id",
      given  = "name/given",
      family = "name/family")))

}, silent = TRUE)
```

## fhir_save *Save FHIR bundles as xml-files*

### Description

Writes a list of FHIR bundles as numbered xml files into a directory.

### Usage

```
fhir_save(bundles, directory)
```

### Arguments

| | |
|---|---|
| bundles | A list of xml objects representing the FHIR bundles. |
| directory | A character vector of length one containing the path to the folder to store the data in. |

### Examples

```
#unserialize example bundle
bundles <- fhir_unserialize(medication_bundles)

#save all bundles to temporary directory
fhir_save(bundles, directory = tempdir())

#save only two bundles (the second and the third) to temporary directory
fhir_save(bundles[c(2,3)], directory = tempdir())
```

## fhir_save_design *Write design to xml*

### Description

Writes a fhir_design for use with `fhir_crack()` to an xml file.

### Usage

```
fhir_save_design(design, file = "design.xml")
```

### Arguments

| | |
|---|---|
| design | A fhir_design object. See `fhir_design()`. |
| file | A string specifying the file to write to, defaults to writing 'design.xml' into the current working directory. |

## See Also

fhir_design(), fhir_table_description(), fhir_load_design()

## Examples

```
#create and save design
table_desc1 <- fhir_table_description(
    resource = 'Patient',
    cols     = c(
        id     = 'id',           # column names with xpaths
        name   = 'name/family',
        gender = 'gender'
    ),
    sep          = ':::',
    brackets     = c('[', ']'),
    rm_empty_cols = FALSE,
    format       = 'compact',
    keep_attr    = FALSE
)

table_desc2 <- fhir_table_description(
    resource = 'Observation',
    cols     = c(
        'code/coding/system', # only xpaths
        'code/coding/code'
    )
)

design <- fhir_design(
    Patients     = table_desc1,
    Observations = table_desc2
)

fhir_save_design(design = design, file = tempfile())
```

---

# fhir_search *Download FHIR search result*

---

## Description

Downloads all FHIR bundles of a FHIR search request from a FHIR server by iterating through the bundles. Search via GET and POST is possible, see Details.

## Usage

```
fhir_search(
  request = fhir_current_request(),
  body = NULL,
  username = NULL,
```

```
  password = NULL,
  token = NULL,
  add_headers = NULL,
  max_bundles = Inf,
  verbose = 1,
  delay_between_attempts = c(1, 3, 9, 27, 81),
  log_errors = NULL,
  save_to_disc = NULL,
  delay_between_bundles = 0,
  rm_tag = "div",
  max_attempts = deprecated()
)
```

## Arguments

| | |
|---|---|
| request | An object of class fhir_url or a character vector of length one containing the full FHIR search request. It is recommended to explicitly create the request via fhir_url() as this will do some validity checks and format the url properly. Defaults to fhir_current_request() |
| body | A character vector of length one or object of class fhir_body with type "application/x-www-form-urle A body should be provided when the FHIR search request is too long and might exceed the maximal allowed length of the URL when send to the server. In this case a search via POST (see https://www.hl7.org/fhir/search.html#Introduction) can be used. The body should contain all the parameters that follow after the ? in the FHIR search request. When a body is provided, the required _search is automatically added to the url in request. See examples and ?fhir_body. |
| username | A character vector of length one containing the username for basic authentication. |
| password | A character vector of length one containing the password for basic authentication. |
| token | A character vector of length one or object of class httr::Token, for bearer token authentication (e.g. OAuth2). See fhir_authenticate() for how to create this. |
| add_headers | A named character vector of custom headers to add to the HTTP request, e.g. c(myHeader = "somevalue") or c(firstHeader = "value1", secondHeader = "value2"). |
| max_bundles | Maximal number of bundles to get. Defaults to Inf meaning all available bundles are downloaded. |
| verbose | An integer vector of length one. If 0, nothing is printed, if 1, only finishing message is printed, if > 1, downloading progress will be printed. Defaults to 1. |
| delay_between_attempts | A numeric vector specifying the delay in seconds between attempts of reaching the server that fhir_search() will make. The length of this vector determines the number of attempts that will be made before stopping with an error. Defaults to c(1,3,9,27,81). |

log_errors — Either NULL or a character vector of length one indicating the name of a file in which to save the http errors. NULL means no error logging. When a file name is provided, the errors are saved in the specified file. Defaults to NULL. Regardless of the value of log_errors the most recent http error message within the current R session is saved internally and can be accessed with fhir_recent_http_error().

save_to_disc — Either NULL or a character vector of length one indicating the name of a directory in which to save the bundles. If a directory name is provided, the bundles are saved as numerated xml-files into the directory specified and not returned as a bundle list in the R session. This is useful when a lot of bundles are to be downloaded and keeping them all in one R session might overburden working memory. When the download is complete, the bundles can be loaded into R using fhir_load(). Defaults to NULL, i.e. bundles are returned as a list within the R session.

delay_between_bundles — A numeric scalar specifying a time in seconds to wait between pages of the search result, i.e. between downloading the current bundle and the next bundle. This can be used to avoid choking a weak server with too many requests to quickly. Defaults to zero.

rm_tag — Character vector of length 1 defining an xml tag of elements that will removed from the bundle automatically. Defaults to "div",leading to the removal of all html parts (see Details). Set to NULL to keep the bundles untouched. See fhir_rm_div() and fhir_rm_tag() for more info.

max_attempts — **[Deprecated]** The number of maximal attempts is now determined by the length of delay_between_attempts

## Details

### Request type:

fhir_search allows for two types of search request:

1. FHIR search via GET: This is the more common approach. All information on which resources to download is contained in the URL that is send to the server (request argument). This encompasses the base url of the server, the resource type and possible search parameters to further qualify the search (see fhir_url()). The search via GET is the default and performed whenever the argument body is NULL.
2. FHIR search via POST: This option should only be used when the parameters make the search URL so long the server might deny it because it exceeds the allowed length. In this case the search parameters (everything that would usually follow the resource type after the ?) can be transferred to a body of type "application/x-www-form-urlencoded" and send via POST. If you provide a body in fhir_search(), the url in request should only contain the base URL and the resource type. The function will automatically amend it with _search and perform a POST.

### Authentication:

There are several ways of authentication implemented in fhir_search(). If you don't need any authentication, just leave the arguments described in the following at their default values of NULL.

1. Basic Authentication: Provide the username and the password for basic authentication in the respective arguments.

2. Token Authentication: Provide a token in the argument `token`, either as a character vector of length one or as as an object of class httr::Token. You can use the function `fhir_authenticate()` to create this object.

**Additional headers:**

Per default, the underlying HTTP requests are equipped with *Accept* and *Authorization* headers. If you need to pass additional headers, e.g. cookies for authentication or other custom headers, you can add these to the request as a named character vector using the `add_headers` argument.

**HTML removal:**

FHIR resources can contain a considerable amount of html code (e.g. in a narrative object), which is often created by the server for example to provide a human-readable summary of the resource. This data is usually not the aim of structured statistical analysis, so in the default setting `fhir_search()` will remove the html parts immediately after download to reduce memory usage (on a hapi server typically by around 30%, see `fhir_rm_div()`). The memory gain is payed with a runtime increase of 10%-20%. The html removal can be disabled by setting `rm_tag = NULL` to increase speed at the cost of increased memory usage.

## Value

A fhir_bundle_list when `save_to_disc = NULL` (the default), else `NULL`.

## See Also

- Creating a FHIR search request: `fhir_url()` and `fhir_body()` (for POST based search)
- OAuth2 Authentication: `fhir_authenticate()`
- Saving/reading bundles from disc: `fhir_save()` and `fhir_load()`
- Flattening the bundles: `fhir_crack()`

## Examples

```
#the try({}, silent = TRUE) statement is only there to catch errors when the server is down
#you can skip it when the server is reachable

try({

### Search with GET

#create fhir search url

request <- fhir_url(url = "https://server.fire.ly",
                    resource = "Patient",
                    parameters = c(gender="female"))

#download bundles
bundles <- fhir_search(request, max_bundles = 5)
```

```
### Search with POST (should actually be used for longer requests)

request <- fhir_url(url = "https://server.fire.ly",
                    resource = "Patient")

body <- fhir_body(content = list(gender = "female"))

bundles <- fhir_search(request = request,
                       body = body,
                       max_bundles = 5)


 }, silent = TRUE)
```

---

`fhir_serialize` *Serialize a fhir_bundle, fhir_bundle_list or fhir_resource*

---

## Description

Serializes FHIR bundles or resources to allow for saving in .rda or .RData format without losing integrity of pointers i.e. it turns a fhir_bundle_xml/fhir_resource_xml object into an fhir_bundle_serialized/fhir_resource_serial object.

## Usage

```
fhir_serialize(bundles)

## S4 method for signature 'fhir_bundle_xml'
fhir_serialize(bundles)

## S4 method for signature 'fhir_bundle_serialized'
fhir_serialize(bundles)

## S4 method for signature 'fhir_bundle_list'
fhir_serialize(bundles)

## S4 method for signature 'fhir_resource_xml'
fhir_serialize(bundles)

## S4 method for signature 'fhir_resource_serialized'
fhir_serialize(bundles)
```

## Arguments

| | |
|---|---|
| bundles | A fhir_bundle, fhir_bundle_list or fhir_resource object. |

## Value

A fhir_bundle_xml, fhir_bundle_list or fhir_resource_xml object.

## Examples

```
#example bundles are serialized, unserialize like this:
bundles <- fhir_unserialize(medication_bundles)

#Serialize like this:
bundles_for_saving <- fhir_serialize(bundles)

#works also on single bundles
fhir_serialize(bundles[[1]])
```

---

# fhir_table_description

*Create fhir_table_description object*

---

## Description

A `fhir_table_description` holds the information `fhir_crack()` needs to flatten (aka crack) FHIR resources from a FHIR bundle. There should be one `fhir_table_description` per resource type as `fhir_crack()` will create one data.frame/data.table per resource type in a bundle. See Details.

## Usage

```
fhir_table_description(
  resource,
  cols = fhir_columns(),
  sep = ":::",
  brackets = character(),
  rm_empty_cols = FALSE,
  format = "compact",
  keep_attr = FALSE,
  style = deprecated()
)
```

## Arguments

| | |
|---|---|
| `resource` | A character vector of length one or fhir_resource_type object indicating which resource type should be extracted. |

| | |
|---|---|
| cols | Optional. A fhir_columns object or something that can be coerced to one, like a (named) character vector, a (named) list containing xpath expressions, or a fhir_xpath_expression object. See fhir_columns() and the examples. If this argument is omitted, an empty fhir_columns object will be supplied. This means that in the call to fhir_crack(), all available elements are extracted in put in automatically named columns. |
| sep | A character of length one containing the separator string used for separating multiple entries in cells when format = "compact". ignored when format = "wide". Defaults to ":::". |
| brackets | A character of length one or two used for the indices of multiple entries. The first one is the opening bracket and the second one the closing bracket. Vectors of length one will be recycled. Defaults to character(0), i.e. no brackets, meaning that multiple entries won't be indexed. |
| rm_empty_cols | A logical of length one indicating whether empty columns should be removed from the resulting table or not. Defaults to FALSE. |
| format | A character of length one indicating whether the resulting table should be cracked to a "wide" or "compact" format. "wide" means multiple entries will be distributed over several columns with indexed names. "compact" means multiple entries will be pasted into one cell/column separated by sep . Defaults to "compact". |
| keep_attr | A logical of length one indicating whether the attribute name of the respective element (@value in most cases) should be attached to the name of the variable in the resulting table. Defaults to FALSE. |
| style | **[Deprecated]** |

## Details

A fhir_table_description consists of the following elements:

- The resource element: Defines the resource type (e.g. Patient or Observation). See fhir_resource_type().
- The cols element: Contains the column names and XPath expressions defining the columns to extract. If this element is empty, fhir_crack() will extract all available elements of the resource and name the columns automatically. See fhir_columns().
- The sep element: A character of length one containing the separator string used for separating multiple entries in cells.
- The brackets element: A character of length one or two used for the indices of multiple entries. The first one is the opening bracket and the second one the closing bracket. Vectors of length one will be recycled. Defaults to character(0), i.e. no brackets, meaning that multiple entries won't be indexed.
- The rm_empty_cols element: A logical of length one indicating whether empty columns should be removed in the resulting table or not. Defaults to FALSE.
- The format element: A character of length one indicating whether the resulting table should be cracked to a wide or compact format. wide means multiple entries will be distributed over several columns with indexed names. compact means multiple entries will be pasted into one cell/column separated by sep . Defaults to compact.

- The keep_attr element: A logical of length one indicating whether the attribute name of the respective element (`@value` in most cases) should be attached to the name of the variable in the resulting table. Defaults to `FALSE`.

A full `fhir_table_description` looks for example like this:

```
fhir_resource_type: Patient

fhir_columns:
column name | xpath expression
------------------------
name        | name/family
gender      | gender
id          | id

sep:            ':::'
brackets:       '[', ']'
rm_empty_cols: FALSE
format:         'compact'
keep_attr:      FALSE
```

## Value

An object of class fhir_table_description.

## Examples

```
# a minimal table description
fhir_table_description(
    resource = "Patient"
)

# named list for cols
fhir_table_description(
    resource = "Patient",
    cols     = list(
        id     = "id",
        name   = "name/family",
        gender = "gender"
    )
)

#unnamed character for cols, colnames are generated automatically
fhir_table_description(
    resource = "Patient",
    cols     = c(
        "id",
        "name/family",
        "gender"
    )
)
```

```
# named character for cols, and overwritten default for other arguments
fhir_table_description(
    resource = "Patient",
    cols = c(
        id            = "id",
        name          = "name/family",
        gender        = "gender"
    ),
    brackets      = c("[", "]"),
    rm_empty_cols = TRUE,
    format        = "wide"
)

# no column arguments is given -> creates a column for all available elements
fhir_table_description(
    resource = "Patient",
    sep           = " <~> ",
    brackets      = c("<<<", ">>>"),
    rm_empty_cols = FALSE,
    format        = "wide"
)
```

---

`fhir_table_description-class`

*A S4 class describing the form of a table produced by* `fhir_crack()`

---

## Description

A `fhir_table_description` holds the information `fhir_crack()` needs to flatten (aka crack) FHIR resources from a FHIR bundle and is created with its constructor function `fhir_table_description()`. Each `fhir_table_description` describes a table for a specific resource type as `fhir_crack()` will create one data.frame/data.table per resource type. See Details.

## Details

A `fhir_table_description` consists of the following elements:

- The resource element: Defines the resource type (e.g. `Patient` or `Observation`). See `fhir_resource_type()`.
- The cols element: Contains the column names and XPath expressions defining the columns to extract. If this element is empty, `fhir_crack()` will extract all available elements of the resource and name the columns automatically. See `fhir_columns()`.
- The sep element: A character of length one containing the separator string used for separating multiple entries in cells.

- The brackets element: A character of length one or two used for the indices of multiple entries. The first one is the opening bracket and the second one the closing bracket. Vectors of length one will be recycled. Defaults to `character(0)`, i.e. no brackets, meaning that multiple entries won't be indexed.
- The rm_empty_cols element: A logical of length one indicating whether empty columns should be removed in the resulting table or not. Defaults to `FALSE`.
- The format element: A character of length one indicating whether the resulting table should be cracked to a `wide` or `compact` format. `wide` means multiple entries will be distributed over several columns with indexed names. `compact` means multiple entries will be pasted into one cell/column separated by `sep` . Defaults to `compact`.
- The keep_attr element: A logical of length one indicating whether the attribute name of the respective element (`@value` in most cases) should be attached to the name of the variable in the resulting table. Defaults to `FALSE`.

A full `fhir_table_description` looks for example like this:

```
fhir_resource_type: Patient

fhir_columns:
column name | xpath expression
------------------------
name        | name/family
gender      | gender
id          | id

sep:           ':::'
brackets:      '[', ']'
rm_empty_cols: FALSE
format:        'compact'
keep_attr:     FALSE
```

## Slots

`resource` An object of class fhir_resource_type defining the resource type that should be extracted.

`cols` An object of class fhir_columns describing which columns should be created and how. If this is an empty fhir_columns object, the call to `fhir_crack()` will extract all available elements and put them in automatically named columns.

`sep` A character of length one containing the separator string used for separating multiple entries in cells when `format = "compact"`. ignored when `format = "wide"`.

`brackets` A character of length one or two used for the indices of multiple entries. The first one is the opening bracket and the second one the closing bracket. Vectors of length one will be recycled. Defaults to `character(0)`, i.e. no brackets, meaning that multiple entries won't be indexed.

`rm_empty_cols` A logical of length one indicating whether empty columns should be removed from the resulting table or not. Defaults to FALSE.

format A character of length one indicating whether the resulting table should be cracked to a wide or compact format. wide means multiple entries will be distributed over several columns with indexed names. compact means multiple entries will be pasted into one cell/column separated by sep . Defaults to compact.

keep_attr A logical of length one indicating whether the attribute name of the respective element (@value in most cases) should be attached to the name of the variable in the resulting table. Defaults to FALSE

## See Also

fhir_resource_type(),fhir_columns(), fhir_design(), fhir_crack()

---

# fhir_tree *Represent a wide cast table as a tree*

---

## Description

This function takes a wide table as created by fhir_crack() with format="wide"and creates the tree structure implicit in the column names of the tables. It is useful to get an overview over the implied structure when planning to create FHIR bundles from this table using fhir_build_bundle().

## Usage

```
fhir_tree(
  table,
  brackets,
  resource = "Resource",
  keep_attr = FALSE,
  keep_ids = FALSE,
  skip_one = FALSE,
  format = "plain",
  prompt = ": "
)
```

## Arguments

| | |
|---|---|
| table | A data.frame or data.table as produced by fhir_crack() with format="wide" or fhir_cast() |
| brackets | A character vector of length two. The brackets used in the table. |
| resource | A character vector of length one or fhir_resource_type object indicating which resource type the table is build from. |
| keep_attr | A logical vector of length one indicating whether attributes should be displayed or not. Only used for formats "plain" and "fancy". |
| keep_ids | A logical vector of length one indicating whether indices should be displayed or not. Only used for formats "plain" and "fancy". |

| | |
|---|---|
| skip_one | A logical vector of length one indicating whether first index 1 should be displayed or not. Only used for formats "plain" and "fancy". |
| format | The format of the tree. One of "plain", "fancy" or "xml". |
| prompt | A character vector of length one use as prompt Only used for formats "plain" and "fancy". |

## Value

A string that can be used with cat() or can be written to a text file.

## See Also

fhir_cast(), fhir_build_bundle()

## Examples

```
#' #unserialize example
bundles <- fhir_unserialize(bundles = example_bundles1)

#crack fhir resources
table_desc <- fhir_table_description(
    resource = "Patient",
    brackets = c("[", "]"),
    sep      = " ",
    format   = "wide"
)

df <- fhir_crack(bundles = bundles, design = table_desc)


###show tree

#plain format
cat(fhir_tree(
       table = df,
       brackets = c("[", "]"),
       resource = "Patient"
    )
 )

#fancy format with indices
cat(fhir_tree(
       table = df,
       brackets = c("[", "]"),
       resource = "Patient",
       format = "fancy",
       keep_ids = TRUE
     )
)

#xml format
cat(fhir_tree(
```

```
      table = df,
      brackets = c("[", "]"),
      resource = "Patient",
      format = "xml"
    )
)
```

---

`fhir_unserialize` *Unserialize a fhir_bundle, fhir_bundle_list or fhir_resource*

---

## Description

Unserializes FHIR resources or bundles that have been serialized to allow for saving in .rda or .RData format, i.e. it turns a fhir_bundle_serialized/fhir_resource_serialized object into an fhir_bundle_xml/fhir_resource_xml object.

## Usage

```
fhir_unserialize(bundles)

## S4 method for signature 'fhir_bundle_xml'
fhir_unserialize(bundles)

## S4 method for signature 'fhir_bundle_serialized'
fhir_unserialize(bundles)

## S4 method for signature 'fhir_resource_xml'
fhir_unserialize(bundles)

## S4 method for signature 'fhir_resource_serialized'
fhir_unserialize(bundles)

## S4 method for signature 'fhir_bundle_list'
fhir_unserialize(bundles)
```

## Arguments

| | |
|---|---|
| `bundles` | A fhir_bundle, fhir_bundle_list or fhir_resource object. |

## Value

A fhir_bundle_serialized, fhir_bundle_list or fhir_resource_serializedobject.

## Examples

```
#unserialize bundle list
fhir_unserialize(patient_bundles)

#unserialize single bundle
fhir_unserialize(patient_bundles[[1]])
```

---

`fhir_url` *Create FHIR URL*

---

## Description

This function creates an object of class fhir_url which mostly represents a URL-encoded URL for a FHIR search request. A valid Search URL contains a base URL and a resource type and may contain additional search parameters. For more info on FHIR search see https://www.hl7.org/fhir/search.html.

## Usage

```
fhir_url(url, resource, parameters, url_enc = TRUE)

## S4 method for signature 'character,missing,missing'
fhir_url(url, url_enc = TRUE)

## S4 method for signature 'character,character,missing'
fhir_url(url, resource, url_enc = TRUE)

## S4 method for signature 'character,character,character'
fhir_url(url, resource, parameters, url_enc = TRUE)

## S4 method for signature 'character,character,list'
fhir_url(url, resource, parameters, url_enc = TRUE)
```

## Arguments

| | |
|---|---|
| `url` | A character of length one specifying either the full search request, e.g. `"http://hapi.fhir.org/baseR4` or the base URL to the FHIR server, e.g. `"http://hapi.fhir.org/baseR4"`. |
| `resource` | A character of length one or fhir_resource_type object with the resource type to be searched, e.g. `"Patient"`. |
| `parameters` | Optional. Either a length 1 character containing properly formatted FHIR search parameters, e.g. `"gender=male&_summary=count"` or a named list or named character vector e.g. `list(gender="male", "_summary"="count")` or `c(gender="male", "_summary"="count")`. Note that parameter names beginning with _ have to be put in quotation marks! |
| `url_enc` | Should the url be URL-encoded? Defaults to `TRUE`. |

## Details

You can use this function in two ways. If you provide just one string in the argument url with the full FHIR search request, this string will be taken as a full FHIR search request. If you also provide the arguments `resource` and/or `parameters`, the string in `url` will be taken as the base url of your FHIR server and the arguments will be concatenated appropriately to form the full request. See examples.

Note that only the latter approach does a validity check on the resource type!

You can disable URL-encoding by setting `url_enc=FALSE`.

## Value

An object of class fhir_url

## Examples

```
#provide full FHIR search request
fhir_url(url = "http://hapi.fhir.org/baseR4/Patient?gender=male&_summary=count")

#provide base url and resource type
fhir_url(
   url      = "http://hapi.fhir.org/baseR4",
   resource = "Patient"
 )

#parameters in one string
fhir_url(
   url        = "http://hapi.fhir.org/baseR4",
   resource   = "Patient",
   parameters = "gender=male&_summary=count"
 )

#parameters as a named character
fhir_url(
   url        = "http://hapi.fhir.org/baseR4",
   resource   = "Patient",
   parameters = c("gender" = "male", "_summary" = "count")
 )

#parameters as a named list
fhir_url(
   url        = "http://hapi.fhir.org/baseR4",
   resource   = "Patient",
   parameters = list("gender" = "male", "_summary" = "count")
 )
```

## fhir_url-class *An S4 object to represent a URL for a FHIR server*

### Description

Objects of this class are basically strings (character vectors of length one) representing a URL. They are usually url encoded. See `fhir_url()` for how to build them.

## fhir_xpath_expression *Create fhir_xpath_expression*

### Description

This function takes a character vector, checks whether it contains valid XPath (1.0) expressions and returns it as an fhir_xpath_expression object. These objects are used in `fhir_parameters` objects.

### Usage

```
fhir_xpath_expression(expression)
```

### Arguments

| | |
|---|---|
| `expression` | A character vector of the XPath expressions |

### Value

A XPath expression object

### Examples

```
fhir_xpath_expression(c("//Patient", "name/given"))
```

## fhir_xpath_expression-class *An S4 class for xpath_expressions Objects of this class are essentially character vectors, but can only be valid XPath (1.0) expressions. They are mostly used in the* `fhir_columns` *class.*

### Description

An S4 class for xpath_expressions Objects of this class are essentially character vectors, but can only be valid XPath (1.0) expressions. They are mostly used in the `fhir_columns` class.

`medication_bundles` *Exemplary FHIR bundles*

## Description

These data examples can be used to explore some of the functions from the fhircrackr package when direct access to a FHIR server is not possible.

All example data sets are fhir_bundle_lists containing fhir_bundle_serialized objects representing FHIR bundles as returned by `fhir_search()`. They have to be unserialized (once per R session), before you can work with them!

## Usage

```
medication_bundles

patient_bundles
```

## Format

An object of class `fhir_bundle_list` of length 3.

An object of class `fhir_bundle_list` of length 2.

## Details

`medication_bundles` contains 3 bundles with MedicationStatement resources representing Medications with Snomed CT code 429374003 and the respective Patient resources that are linked to these MedicationStatements.

`patient_bundles` contains 2 bundles with Patient resources.

## Source

The data sets are generated by the following code:

**medication_bundles** (*Downloaded 10-05-21*)

```
search_request  <- fhir_url(url = "https://hapi.fhir.org/baseR4",
resource = "MedicationStatement",
parameters = c("code" = "http://snomed.info/ct|429374003",
   "_include" = "MedicationStatement:subject"))

bundles <- fhir_search(request = search_request, max_bundles = 3)

medication_bundles <- fhir_serialize(bundles = bundles)
```

**patient_bundles** (*Downloaded 10-05-21*)

```
bundles <- fhir_search(request="https://hapi.fhir.org/baseR4/Patient",
                       max_bundles=2,
                       verbose = 0)

patient_bundles <- fhir_serialize(bundles = bundles)
```

## Examples

```
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = medication_bundles)

#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = patient_bundles)
```

---

# pastep *Concatenate paths*

---

## Description

Concatenates two or more strings to a path string correctly.

## Usage

```
pastep(..., list_of_paths = NULL, ext = NULL)
```

## Arguments

| | |
|---|---|
| `...` | A Set of Path Strings. Only works if list_of_paths is NULL |
| `list_of_paths` | Either a vector or a list of paths strings |
| `ext` | An Extension to add at the end of the path |

## Value

A Character of length one, the combined path.

## Examples

```
pastep('a', 'b', 'c', 'd')
pastep(list_of_paths = list(paste0('/', letters, '/'), as.character(1:13)))
pastep(list_of_paths = c(letters, as.character(1:13)))
pastep(list_of_paths = c(letters, as.character(1:13)), ext = '.txt')
pastep(list_of_paths = c(letters, as.character(1:13)), ext = '_dat.txt')
```

## paste_paths *Concatenate two paths*

### Description

Concatenates two strings to a path string correctly.

### Usage

```
paste_paths(path1 = "w", path2 = "d", os = "LiNuX")
```

### Arguments

| | |
|---|---|
| path1 | A a character vector of length one specifying the left hand part of the resulting path. |
| path2 | A a character vector of length one specifying the right hand part of the resulting path. |
| os | A a character vector of length one specifying the operating system you're operating on: windows or linux. |

### Value

A a character vector of length one containing the concatenated path.

### Examples

```
paste_paths(path1 = "data", path2 = "patients")
paste_paths(path1 = "/data", path2 = "patients")
paste_paths(path1 = "/data/", path2 = "patients")
paste_paths(path1 = "/data", path2 = "/patients")
paste_paths(path1 = "/data/", path2 = "/patients/")
paste_paths(path1 = "data", path2 = "patients", os = "windows")
```

## transaction_bundle_example *Toy examples to POST/PUT on a server*

### Description

These data examples are simple examples to try out POSTing/PUTing resources to a server. See **Source** for how the xml versions look.

## Usage

```
transaction_bundle_example

example_resource1

example_resource2

example_resource3
```

## Format

An object of class `fhir_bundle_serialized` of length 1277.

An object of class `fhir_resource_serialized` of length 267.

An object of class `fhir_resource_serialized` of length 290.

An object of class `fhir_resource_serialized` of length 608.

## Details

`transaction_bundle_example` contains 1 transaction bundle with 2 Patient resources.

`example_resource1` contains 1 patient resource without id for POSTing

`example_resource2` contains 1 patient resource with id for PUTing

`example_resource3` contains 1 Medication resource with an id xml attribute

## Source

**transaction_bundle_example**

```
<Bundle>
   <type value='transaction'/>
  <entry>
     <resource>
        <Patient>
           <id value='id1'/>
           <address>
              <use value='home'/>
              <city value='Amsterdam'/>
              <type value='physical'/>
              <country value='Netherlands'/>
           </address>
           <name>
              <given value='Marie'/>
           </name>
        </Patient>
     </resource>
     <request>
        <method value='POST'/>
        <url value='Patient'/>
```

```
    </request>
  </entry>
  <entry>
      <resource>
         <Patient>
            <id value='id3'/>
            <address>
               <use value='home'/>
               <city value='Berlin'/>
            </address>
            <address>
               <use value='work'/>
               <city value='London'/>
               <type value='postal'/>
               <country value='England'/>
             </address>
            <address>
               <type value='postal'/>
               <country value='France'/>
            </address>
            <name>
               <given value='Frank'/>
            </name>
            <name>
               <given value='Max'/>
            </name>
         </Patient>
      </resource>
      <request>
         <method value='POST'/>
         <url value='Patient'/>
      </request>
  </entry>
</Bundle>
```

**example_resource1**

```
<Patient>
   <name>
     <given value = 'Marie'/>
  </name>
  <gender value = 'female'/>
  <birthDate value = '1970-01-01'/>
</Patient>
```

**example_resource2**

```
<Patient>
```

```
   <id value = '1a2b3c'/>
   <name>
     <given value = 'Marie'/>
  </name>
  <gender value = 'female'/>
  <birthDate value = '1970-01-01'/>
</Patient>
```

**example_resource3**

```
<Medication>
    <code>
        <coding>
            <system value="http://www.nlm.nih.gov/research/umls/rxnorm"/>
            <code value="1594660"/>
            <display value="Alemtuzumab 10mg/ml (Lemtrada)"/>
        </coding>
    </code>
    <ingredient id="1">
     <itemReference>
      <reference value="Substance/5463"/>
    </itemReference>
    </ingredient>
    <ingredient id="2">
     <itemReference>
      <reference value="Substance/3401"/>
  </itemReference>
    </ingredient>
</Medication>
```

## Examples

```
#unserialize xml objects before doing anything else with them!
fhir_unserialize(bundles = transaction_bundle_example)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(example_resource1)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(example_resource2)
#unserialize xml objects before doing anything else with them!
fhir_unserialize(example_resource3)
```

# Index


∗ **datasets**
example_bundles1, 4
medication_bundles, 80
transaction_bundle_example, 82

as_fhir, 3

example_bundles1, 4
example_bundles2 *(*example_bundles1*)*, 4
example_bundles3 *(*example_bundles1*)*, 4
example_bundles4 *(*example_bundles1*)*, 4
example_bundles5 *(*example_bundles1*)*, 4
example_resource1 *(*transaction_bundle_example*)*, 82
example_resource2 *(*transaction_bundle_example*)*, 82
example_resource3 *(*transaction_bundle_example*)*, 82

fhir_authenticate, 12
fhir_authenticate(), *22*, *28*, *38*, *40*, *47*, *49*, *59*, *61*, *65*, *67*
fhir_body, *13*, 13, *14*, *47*, *49*
fhir_body(), *14*, *47*, *49*, *67*
fhir_body,character,character-method *(*fhir_body*)*, 13
fhir_body,character,character-methods *(*fhir_body*)*, 13
fhir_body,list,character-method *(*fhir_body*)*, 13
fhir_body,list,character-methods *(*fhir_body*)*, 13
fhir_body,list,missing-method *(*fhir_body*)*, 13
fhir_body,list,missing-methods *(*fhir_body*)*, 13
fhir_body-class, 14
fhir_build_bundle, 14
fhir_build_bundle(), *15, 16*, *18*, *24*, *47*, *74, 75*
fhir_build_bundle,data.frame-method *(*fhir_build_bundle*)*, 14
fhir_build_bundle,list-method *(*fhir_build_bundle*)*, 14
fhir_build_resource, 17
fhir_build_resource(), *16*, *47*, *49*
fhir_bundle, *19*, *68*, *76*
fhir_bundle-class, 18
fhir_bundle_list, *3*, *19*, 19, *29*, *38*, *41*, *54*, *56, 57*, *60*, *67–69*, *76*, *80*
fhir_bundle_list-class, 20
fhir_bundle_serialized, *68*, *76*, *80*
fhir_bundle_serialized-class, 20
fhir_bundle_xml, *14*, *16*, *20*, 20, *47*, *54*, *56, 57*, *68, 69*, *76*
fhir_bundle_xml-class, 21
fhir_canonical_design, 21
fhir_capability_statement, 22
fhir_cast, 23
fhir_cast(), *16*, *18*, *24*, *31*, *74, 75*
fhir_columns, *25*, 25, *70*, *73*
fhir_columns(), *70*, *72*, *74*
fhir_columns,character,character-method *(*fhir_columns*)*, 25
fhir_columns,character,missing-method *(*fhir_columns*)*, 25
fhir_columns,list,missing-method *(*fhir_columns*)*, 25
fhir_columns,missing,missing-method *(*fhir_columns*)*, 25
fhir_columns,NULL,missing-method *(*fhir_columns*)*, 25
fhir_columns-class, 26
fhir_common_columns, 26
fhir_common_columns(), *44*
fhir_count_resource, 27

fhir_count_resource(), *59, 60*, *62*
fhir_crack, 29
fhir_crack(), *14–18*, *21*, *23, 24*, *26*, *33, 34*, *36*, *41*, *43*, *53*, *55, 56*, *63*, *67*, *69, 70*, *72–74*
fhir_crack,ANY,fhir_design-method *(fhir_crack)*, 29
fhir_crack,ANY,fhir_table_description-method *(fhir_crack)*, 29
fhir_crack,fhir_design-method *(fhir_crack)*, 29
fhir_crack,fhir_table_description-method *(fhir_crack)*, 29
fhir_current_request, 32
fhir_current_request(), *65*
fhir_design, *14*, *17*, *21*, *26*, *30, 31*, *33*, 33, *36*, *41, 42*, *63*
fhir_design(), *21*, *24*, *26*, *30, 31*, *42*, *63, 64*, *74*
fhir_design,fhir_table_description-method *(fhir_design)*, 33
fhir_design,fhir_table_list-method *(fhir_design)*, 33
fhir_design,list-method *(fhir_design)*, 33
fhir_design-class, 36
fhir_df_list, *31*, *33, 34*
fhir_df_list-class, 36
fhir_dt_list, *31*, *33, 34*
fhir_dt_list-class, 36
fhir_get_resource_ids, 39
fhir_get_resource_ids(), *38*, *59*
fhir_get_resources_by_ids, 37
fhir_get_resources_by_ids(), *40*, *59–62*
fhir_load, 40
fhir_load(), *66, 67*
fhir_load_design, 41
fhir_load_design(), *64*
fhir_melt, 43
fhir_melt(), *22*, *24*, *26, 27*, *31*, *56*
fhir_next_bundle_url, 44
fhir_post, 46
fhir_post(), *16*, *18*, *47*
fhir_post,ANY,fhir_body-method *(fhir_post)*, 46
fhir_post,ANY,fhir_bundle_xml-method *(fhir_post)*, 46
fhir_post,ANY,fhir_resource-method *(fhir_post)*, 46
fhir_post,fhir_body-method *(fhir_post)*, 46
fhir_post,fhir_bundle_xml-method *(fhir_post)*, 46
fhir_post,fhir_resource-method *(fhir_post)*, 46
fhir_put, 48
fhir_put(), *18*, *49*
fhir_recent_http_error, 50
fhir_recent_http_error(), *47*, *49*, *66*
fhir_request, 51
fhir_resource, *47*, *49*, *68*, *76*
fhir_resource-class, 52
fhir_resource_serialized, *68*, *76*
fhir_resource_serialized-class, 52
fhir_resource_type, *15*, *18*, *28*, *37*, *39*, *52*, 52, *53*, *59*, *61*, *69*, *73, 74*, *77*
fhir_resource_type(), *70*, *72*, *74*
fhir_resource_type-class, 53
fhir_resource_xml, *17, 18*, *53*, 53, *68, 69*, *76*
fhir_resource_xml,character-method *(fhir_resource_xml)*, 53
fhir_resource_xml,xml_document-method *(fhir_resource_xml)*, 53
fhir_resource_xml,xml_node-method *(fhir_resource_xml)*, 53
fhir_resource_xml-class, 54
fhir_rm_div, 54
fhir_rm_div(), *57*, *66, 67*
fhir_rm_indices, 55
fhir_rm_indices(), *27*, *31*, *44*
fhir_rm_tag, 56
fhir_rm_tag(), *54*, *66*
fhir_rm_tag,character-method *(fhir_rm_tag)*, 56
fhir_rm_tag,fhir_bundle_list-method *(fhir_rm_tag)*, 56
fhir_rm_tag,fhir_bundle_xml-method *(fhir_rm_tag)*, 56
fhir_sample_resources, 58
fhir_sample_resources(), *62*
fhir_sample_resources_by_ids, 60
fhir_sample_resources_by_ids(), *59, 60*
fhir_save, 63
fhir_save(), *41*, *67*
fhir_save_design, 63
fhir_save_design(), *42*

fhir_search, *12*, 64
fhir_search(), *15*, *19–21*, *28–32*, *38*, *40*, *44*, *45*, *50*, *59–62*, *67*, *80*
fhir_serialize, 68
fhir_serialize(), *20*, *52*
fhir_serialize,fhir_bundle_list-method (fhir_serialize), 68
fhir_serialize,fhir_bundle_serialized-method (fhir_serialize), 68
fhir_serialize,fhir_bundle_xml-method (fhir_serialize), 68
fhir_serialize,fhir_resource_serialized-method (fhir_serialize), 68
fhir_serialize,fhir_resource_xml-method (fhir_serialize), 68
fhir_table_description, *14*, *17*, *26*, *30*, *31*, *36*, *69*, 69, *71*
fhir_table_description(), *21*, *24*, *30*, *31*, *33*, *34*, *36*, *42*, *64*, *72*
fhir_table_description-class, 72
fhir_tree, 74
fhir_unserialize, 76
fhir_unserialize,fhir_bundle_list-method (fhir_unserialize), 76
fhir_unserialize,fhir_bundle_serialized-method (fhir_unserialize), 76
fhir_unserialize,fhir_bundle_xml-method (fhir_unserialize), 76
fhir_unserialize,fhir_resource_serialized-method (fhir_unserialize), 76
fhir_unserialize,fhir_resource_xml-method (fhir_unserialize), 76
fhir_url, *21*, *45*, *47*, *49*, *65*, *77*, 77, *78*
fhir_url(), *32*, *65–67*, *79*
fhir_url,character,character,character-method (fhir_url), 77
fhir_url,character,character,list-method (fhir_url), 77
fhir_url,character,character,missing-method (fhir_url), 77
fhir_url,character,missing,missing-method (fhir_url), 77
fhir_url-class, 79
fhir_xpath_expression, *25*, *70*, 79
fhir_xpath_expression-class, 79

httr::oauth2.0_token(), *12*
httr::oauth_app(), *12*
httr::oauth_endpoint(), *12*
httr::POST(), *46*
httr::PUT(), *48*
httr::Token, *12*, *22*, *28*, *38*, *40*, *47*, *49*, *59*, *61*, *65*, *67*

medication_bundles, 80

paste_paths, 82
pastep, 81
patient_bundles (medication_bundles), 80

transaction_bundle_example, 82