Package: orthanc (via r-universe)

July 2, 2026

Title Programmatic Interface to 'Orthanc' DICOM Servers

Version 0.3.0

Description An R Interface to 'Orthanc' DICOM servers for medical imaging workflows. 'Orthanc' is a lightweight, open-source DICOM server that exposes a comprehensive REST API for managing, querying, retrieving, and modifying DICOM resources (<https://www.orthanc-server.com>). The goal of this package is to provide comprehensive and user-friendly access to the 'Orthanc' REST API, designed to align with idiomatic R workflows while preserving the structure and semantics of DICOM resources.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3.9000

Depends R (>= 4.1.0)

Imports carrier (>= 0.3.0), clock, digest, glue, fs, httr2, jsonlite, mirai (>= 2.5.1), prettyunits, purrr (>= 1.1.0), R6, rlang

Suggests png, jpeg

URL https://github.com/mattwarkentin/orthanc, https://mattwarkentin.github.io/orthanc/

BugReports https://github.com/mattwarkentin/orthanc/issues

NeedsCompilation no

Author Matthew T. Warkentin [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-8730-3511>)

Maintainer Matthew T. Warkentin <matthew.warkentin@ucalgary.ca>

Repository https://cran.r-universe.dev

Date/Publication 2026-07-01 23:20:02 UTC

RemoteUrl https://github.com/cran/orthanc

RemoteRef HEAD

RemoteSha 327762541e1241b8df4b2c68d1c693c42d8e4cbc

Contents

`find_and_filter_patients` *Find and filter patients using predicate functions*

Description

Find desired Patient/Study/Series/Instance in an Orthanc server. Predicate functions (filters) take a single Patient/Study/Series/Instance as the first argument and return a single `TRUE` or `FALSE` for whether the resource should be kept or discarded, respectively. If `mirai::daemons()` has been used to set persistent background processes, this function will apply filters in parallel (over patients) using all available processes.

Usage

```
find_and_filter_patients(
  client,
  patient_filter = NULL,
  study_filter = NULL,
  series_filter = NULL,
  instance_filter = NULL,
  progress = rlang::is_interactive(),
  ...
)
```

Arguments

client	Orthanc client.
patient_filter	Predicate function to filter Patients.
study_filter	Predicate function to filter Studys.
series_filter	Predicate function to filer Series.
instance_filter	Predicate function to filter Instances.
progress	Whether to show progress bars. By default, progress bars are enabled in interactive sessions (i.e., if rlang::is_interactive() returns TRUE).
...	Named-arguments to declare in the environment of the predicate functions, if performing filtering in parallel with mirai. If your predicate functions require access to objects in the global environment,] you must pass them to ... so they are available on each background worker when running in parallel (i.e., when mirai::daemons() has been set).

Details

This function builds a series of tree structures. Each tree corresponds to a patient. The layers in the tree correspond to:

Patient -> Studies -> Series -> Instances

Value

A list of filtered Patients.

Examples

```
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")

find_and_filter_patients(
  client = client,
  patient_filter = \(pt) pt$is_stable
)
```

---

find_instances *Finds instances in Orthanc according to queries and labels*

Description

Finds instances in Orthanc according to queries and labels

Usage

```
find_instances(
  client,
  query = list(),
  labels = character(),
  labels_constraint = "All",
  ...
)
```

Arguments

client	Orthanc API client.
query	Named-list that specifies the filters on the level related DICOM tags.
labels	Character vector of labels to look for in resources.
labels_constraint	Contraint on the labels ('All', 'Any', 'None').
...	Additional arguments passed to query_orthanc.

Value

A list of Instance objects.

Examples

```
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")
find_instances(
  client = client,
 query = list(SOPInstanceUID = "1.3.6.1.4.1.14519.5.2.1.2193.7172.260209224923274040650639981398")
)
```

---

find_patients *Finds patients in Orthanc according to queries and labels*

Description

Finds patients in Orthanc according to queries and labels

Usage

```
find_patients(
  client,
  query = list(),
  labels = character(),
  labels_constraint = "All",
  ...
)
```

Arguments

client	Orthanc API client.
query	Named-list that specifies the filters on the level related DICOM tags.
labels	Character vector of labels to look for in resources.
labels_constraint	Contraint on the labels ('All', 'Any', 'None').
...	Additional arguments passed to query_orthanc.

Value

A list of Patient objects.

Examples

```
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")
find_patients(client, query = list(PatientName = "HN_P001"))
```

---

find_series *Finds series in Orthanc according to queries and labels*

Description

Finds series in Orthanc according to queries and labels

Usage

```
find_series(
  client,
  query = list(),
  labels = character(),
  labels_constraint = "All",
  ...
)
```

Arguments

client	Orthanc API client.
query	Named-list that specifies the filters on the level related DICOM tags.
labels	Character vector of labels to look for in resources.
labels_constraint	Contraint on the labels ('All', 'Any', 'None').
...	Additional arguments passed to query_orthanc.

Value

A list of Series objects.

Examples

```
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")
find_series(client, query = list(SeriesDescription = "HEAD/NECK  2.0  B30s"))
```

`find_studies` *Finds studies in Orthanc according to queries and labels*

Description

Finds studies in Orthanc according to queries and labels

Usage

```
find_studies(
  client,
  query = list(),
  labels = character(),
  labels_constraint = "All",
  ...
)
```

Arguments

`client`	Orthanc API client.
`query`	Named-list that specifies the filters on the level related DICOM tags.
`labels`	Character vector of labels to look for in resources.
`labels_constraint`	Contraint on the labels ('All', 'Any', 'None').
`...`	Additional arguments passed to `query_orthanc`.

Value

A `list` of Study objects.

Examples

```
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")
find_studies(client, query = list(StudyDescription = "RT^HEAD_NECK (Adult)"))
```

`Instance` *DICOM Instance Class*

Description

An abstract class for a DICOM Instance resource.

Value

An R6 instance of class `"Instance"`.

Super class

`orthanc::Resource` -> `Instance`

Active bindings

`uid` SOPInstanceUID
`file_size` File size
`creation_date` Creation Date
`series_identifier` Parent series identifier
`parent_series` Parent series
`parent_study` Parent study
`parent_patient` Parent patient
`acquisition_number` Acquisition Number
`image_index` Image Index
`image_orientation_patient` Image Orientation Patient
`image_position_patient` Image Position Patient
`image_comments` Image Comments
`instance_number` Instance Number
`temporal_position_identifier` Temporal Position Identifier
`tags` Tags
`simplified_tags` Simplified Tags
`labels` Get or add labels
`statistics` Statistics
`list_frames` List frames (zero-based indexing)
`frames` Number of frames
`rows` Number of rows
`columns` Number of columns
`shape` Shape (rows x columns x frames (if multi-frame))
`dim` Number of dimensions
`slice_thickness` Slice thickness
`pixel_spacing` Pixel spacing

Methods

Public methods:

- Instance$get_dicom_file_content()
- Instance$download_dicom()
- Instance$get_main_information()
- Instance$add_label()
- Instance$has_label()
- Instance$remove_label()
- Instance$get_raw_content_by_tag()
- Instance$anonymize()
- Instance$modify()
- Instance$get_nifti_file_content()
- Instance$download_nifti()
- Instance$download_image()
- Instance$clone()

Method `get_dicom_file_content()`**:** Retrieves bytes of DICOM file content
This method retrieves bytes corresponding to the DICOM file.

Usage:

```
Instance$get_dicom_file_content()
```

Method `download_dicom()`**:** Download DICOM file to a path.

Usage:

```
Instance$download_dicom(path, stream = FALSE)
```

Arguments:

`path` Path on disk.

`stream` Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.

Method `get_main_information()`**:** Get instance information.

Usage:

```
Instance$get_main_information()
```

Method `add_label()`**:** Add label to resource.

Usage:

```
Instance$add_label(label)
```

Arguments:

`label` Label.

Method `has_label()`**:** Test if resource has label.

Usage:

```
Instance$has_label(label)
```

Arguments:

`label` Label.

Method `remove_label()`**:** Delete label from resource.

Usage:

```
Instance$remove_label(label)
```

Arguments:

`label` Label.

Method `get_raw_content_by_tag()`**:** Get raw content of one DICOM tag.

Usage:

```
Instance$get_raw_content_by_tag(tag)
```

Arguments:

`tag` tag.

Method `anonymize()`**:** Anonymize Instance

Usage:

```
Instance$anonymize(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.
`replace` Named-list of tags to replce.
`keep` List of tags to keep unchanged.
`keep_private_tags` Keep private tags from DICOM instance.
`keep_source` Keep original resource.
`private_creator` Private creator to be used for private tags in replace.
`force` Force tags to be changed.
`dicom_version` Version of the DICOM standard to use for anonymization.

Method `modify()`**:** Modify an Instance

Usage:

```
Instance$modify(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `get_nifti_file_content()`**:** Get bytes of NIfTI file content.

Usage:

```
Instance$get_nifti_file_content(compress = TRUE)
```

Arguments:

`compress` Compress to gzip (nii.gz) Default is `TRUE`.

Method `download_nifti()`**:** Download instance as NIfTI.

Usage:

```
Instance$download_nifti(path, compress = TRUE, stream = FALSE)
```

Arguments:

`path` Path on disk.

`compress` Compress to gzip (nii.gz) Default is `TRUE`.

`stream` Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.

Method `download_image()`**:** Download instance as an image.

Usage:

```
Instance$download_image(
  path,
  frame = 0L,
  format = c("png", "jpeg"),
  render = TRUE,
  params = NULL,
  ...
)
```

Arguments:

`path` Path on disk.

`frame` Index of the frame (starts at `0L`). Default is `0L`.

`format` One of `"png"` or `"jpeg"`. Default is `"png"`.

`render` Should the image be scaled (with `RescaleSlope` and `RescaleIntercept`) and windowed (with `WindowCenter` and `WindowWidth`, if available). Default is `TRUE`.

`params` Optional named-list of query parameters.

`...` Optional arguments passed on to `png::writePNG()` or `jpeg::writeJPEG()`.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Instance$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Job` *Job Class*

Description

Job class to follow a Job in Orthanc

Value

An instance of `Job`.

Active bindings

`state` Job state.

`content` Job content.

`type` Job type.

`creation_time` Job creation time.

`effective_runtime` Job effective runtime.

`priority` Job priority.

`progress` Job progress.

`error` Job error.

`error_details` Job error details.

`timestamp` Job timestamp.

`completion_time` Job completion time.

Methods

Public methods:

- `Job$new()`
- `Job$wait_until_completion()`
- `Job$get_information()`
- `Job$clone()`

Method `new()`**:** Create a new Job instance.

Usage:

```
Job$new(id, client)
```

Arguments:

`id` Job ID.
`client` Orthanc API client.

Method `wait_until_completion()`**:** Stop execution until job is not Pending/Running.

Usage:

```
Job$wait_until_completion(interval = 2L)
```

Arguments:

`interval` Time interval to check the job status, default is 2s.

Method `get_information()`**:** Get job information.

Usage:

```
Job$get_information()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Job$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Modality` *Modality Class*

Description

Wrapper around Orthanc API when dealing with a modality

Value

An instance of `Modality`.

Public fields

`modality` Modality.

Methods

Public methods:

- `Modality$initalize()`
- `Modality$echo()`
- `Modality$get()`
- `Modality$find()`
- `Modality$move()`
- `Modality$store()`
- `Modality$get_query_answers()`

- Modality$clone()

Method `initalize()`**:** Create a new Modality instance.

Usage:

```
Modality$initalize(client, modality)
```

Arguments:

`client` Orthanc API client.

`modality` Remote modality.

Method `echo()`**:** C-Echo to modality

Usage:

```
Modality$echo()
```

Method `get()`**:** C-Move SCU: Send all the results to another modality whose AET is in the body.

Usage:

```
Modality$get(level, resources)
```

Arguments:

`level` Level of the query ("Patient", "Study", "Series", "Instance").

`resources` List or named-list of DICOM tags that identify data to retrieve (e.g., list(StudyInstanceUID = "1.3.6.1.4.1.22213.2.6291.2.1")).

Method `find()`**:** C-Find (Querying with data)

Usage:

```
Modality$find(data)
```

Arguments:

`data` Named-list to send in the body of request.

Method `move()`**:** C-Find (Querying with data)

Usage:

```
Modality$move(query_id, cmove_data)
```

Arguments:

`query_id` Query identifier.

`cmove_data` Ex. list(TargetAet = 'target_modality_name', Synchronous = FALSE)

Method `store()`**:** Store series or instance to modality.

Usage:

```
Modality$store(instance_or_series_id)
```

Arguments:

`instance_or_series_id` Instance or Series Orthanc identifier.

Method `get_query_answers()`**:** Get query answers.

Usage:

```
Modality$get_query_answers(query_id)
```

Arguments:

`query_id` Query identifier.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Modality$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Orthanc` *Orthanc API Client*

Description

Orthanc is an open-source, lightweight DICOM server for healthcare and medical research. This `R6` generator creates a client to access Orthanc's RESTful API. More details about the Orthanc REST API can be found here: https://orthanc.uclouvain.be/book/users/rest.html.

The full documentation of the Orthanc API can be found here: https://orthanc.uclouvain.be/api/.

Value

An `Orthanc` instance.

Public fields

`url` URL for Orthanc REST API.

`api_version` Orthanc REST API version.

Active bindings

`num_instances` Number of instances.

`num_series` Number of series.

`num_studies` Number of studies.

`num_patients` Number of patients.

Methods

Public methods:

- Orthanc$new()
- Orthanc$GET()
- Orthanc$DELETE()
- Orthanc$POST()
- Orthanc$PUT()
- Orthanc$stream()
- Orthanc$print()
- Orthanc$delete_changes()
- Orthanc$get_changes()
- Orthanc$delete_exports()
- Orthanc$get_exports()
- Orthanc$get_instances()
- Orthanc$post_instances()
- Orthanc$delete_instances_id()
- Orthanc$get_instances_id()
- Orthanc$post_instances_id_anonymize()
- Orthanc$get_instances_id_attachments()
- Orthanc$delete_instances_id_attachments_name()
- Orthanc$get_instances_id_attachments_name()
- Orthanc$put_instances_id_attachments_name()
- Orthanc$post_instances_id_attachments_name_compress()
- Orthanc$get_instances_id_attachments_name_compressed_data()
- Orthanc$get_instances_id_attachments_name_compressed_md5()
- Orthanc$get_instances_id_attachments_name_compressed_size()
- Orthanc$get_instances_id_attachments_name_data()
- Orthanc$get_instances_id_attachments_name_info()
- Orthanc$get_instances_id_attachments_name_is_compressed()
- Orthanc$get_instances_id_attachments_name_md5()
- Orthanc$get_instances_id_attachments_name_size()
- Orthanc$post_instances_id_attachments_name_uncompress()
- Orthanc$post_instances_id_attachments_name_verify_md5()
- Orthanc$get_instances_id_content_path()
- Orthanc$post_instances_id_export()
- Orthanc$get_instances_id_file()
- Orthanc$get_instances_id_frames()
- Orthanc$get_instances_id_frames_frame()
- Orthanc$get_instances_id_frames_frame_image_int16()
- Orthanc$get_instances_id_frames_frame_image_uint16()
- Orthanc$get_instances_id_frames_frame_image_uint8()
- Orthanc$get_instances_id_frames_frame_matlab()

- Orthanc$get_instances_id_frames_frame_numpy()
- Orthanc$get_instances_id_frames_frame_preview()
- Orthanc$get_instances_id_frames_frame_raw()
- Orthanc$get_instances_id_frames_frame_raw.gz()
- Orthanc$get_instances_id_frames_frame_rendered()
- Orthanc$get_instances_id_header()
- Orthanc$get_instances_id_image_int16()
- Orthanc$get_instances_id_image_uint16()
- Orthanc$get_instances_id_image_uint8()
- Orthanc$get_instances_id_labels()
- Orthanc$delete_instances_id_labels_label()
- Orthanc$get_instances_id_labels_label()
- Orthanc$put_instances_id_labels_label()
- Orthanc$get_instances_id_matlab()
- Orthanc$get_instances_id_metadata()
- Orthanc$delete_instances_id_metadata_name()
- Orthanc$get_instances_id_metadata_name()
- Orthanc$put_instances_id_metadata_name()
- Orthanc$post_instances_id_modify()
- Orthanc$get_instances_id_module()
- Orthanc$get_instances_id_numpy()
- Orthanc$get_instances_id_patient()
- Orthanc$get_instances_id_pdf()
- Orthanc$get_instances_id_preview()
- Orthanc$post_instances_id_reconstruct()
- Orthanc$get_instances_id_rendered()
- Orthanc$get_instances_id_series()
- Orthanc$get_instances_id_simplified_tags()
- Orthanc$get_instances_id_statistics()
- Orthanc$get_instances_id_study()
- Orthanc$get_instances_id_tags()
- Orthanc$get_jobs()
- Orthanc$delete_jobs_id()
- Orthanc$get_jobs_id()
- Orthanc$post_jobs_id_cancel()
- Orthanc$post_jobs_id_pause()
- Orthanc$post_jobs_id_resubmit()
- Orthanc$post_jobs_id_resume()
- Orthanc$delete_jobs_id_key()
- Orthanc$get_jobs_id_key()
- Orthanc$get_modalities()
- Orthanc$delete_modalities_id()

- Orthanc$get_modalities_id()
- Orthanc$put_modalities_id()
- Orthanc$get_modalities_id_configuration()
- Orthanc$post_modalities_id_echo()
- Orthanc$post_modalities_id_find_worklist()
- Orthanc$post_modalities_id_get()
- Orthanc$post_modalities_id_move()
- Orthanc$post_modalities_id_query()
- Orthanc$post_modalities_id_storage_commitment()
- Orthanc$post_modalities_id_store()
- Orthanc$post_modalities_id_store_straight()
- Orthanc$get_patients()
- Orthanc$delete_patients_id()
- Orthanc$get_patients_id()
- Orthanc$post_patients_id_anonymize()
- Orthanc$get_patients_id_archive()
- Orthanc$post_patients_id_archive()
- Orthanc$get_patients_id_attachments()
- Orthanc$delete_patients_id_attachments_name()
- Orthanc$get_patients_id_attachments_name()
- Orthanc$put_patients_id_attachments_name()
- Orthanc$post_patients_id_attachments_name_compress()
- Orthanc$get_patients_id_attachments_name_compressed_data()
- Orthanc$get_patients_id_attachments_name_compressed_md5()
- Orthanc$get_patients_id_attachments_name_compressed_size()
- Orthanc$get_patients_id_attachments_name_data()
- Orthanc$get_patients_id_attachments_name_info()
- Orthanc$get_patients_id_attachments_name_is_compressed()
- Orthanc$get_patients_id_attachments_name_md5()
- Orthanc$get_patients_id_attachments_name_size()
- Orthanc$post_patients_id_attachments_name_uncompress()
- Orthanc$post_patients_id_attachments_name_verify_md5()
- Orthanc$get_patients_id_instances()
- Orthanc$get_patients_id_instances_tags()
- Orthanc$get_patients_id_labels()
- Orthanc$delete_patients_id_labels_label()
- Orthanc$get_patients_id_labels_label()
- Orthanc$put_patients_id_labels_label()
- Orthanc$get_patients_id_media()
- Orthanc$post_patients_id_media()
- Orthanc$get_patients_id_metadata()
- Orthanc$delete_patients_id_metadata_name()

- Orthanc$get_patients_id_metadata_name()
- Orthanc$put_patients_id_metadata_name()
- Orthanc$post_patients_id_modify()
- Orthanc$get_patients_id_module()
- Orthanc$get_patients_id_protected()
- Orthanc$put_patients_id_protected()
- Orthanc$post_patients_id_reconstruct()
- Orthanc$get_patients_id_series()
- Orthanc$get_patients_id_shared_tags()
- Orthanc$get_patients_id_statistics()
- Orthanc$get_patients_id_studies()
- Orthanc$get_peers()
- Orthanc$delete_peers_id()
- Orthanc$get_peers_id()
- Orthanc$put_peers_id()
- Orthanc$get_peers_id_configuration()
- Orthanc$post_peers_id_store()
- Orthanc$post_peers_id_store_straight()
- Orthanc$get_peers_id_system()
- Orthanc$get_plugins()
- Orthanc$get_plugins_explorer.js()
- Orthanc$get_plugins_id()
- Orthanc$get_queries()
- Orthanc$delete_queries_id()
- Orthanc$get_queries_id()
- Orthanc$get_queries_id_answers()
- Orthanc$get_queries_id_answers_index()
- Orthanc$get_queries_id_answers_index_content()
- Orthanc$post_queries_id_answers_index_query_instances()
- Orthanc$post_queries_id_answers_index_query_series()
- Orthanc$post_queries_id_answers_index_query_studies()
- Orthanc$post_queries_id_answers_index_retrieve()
- Orthanc$get_queries_id_level()
- Orthanc$get_queries_id_modality()
- Orthanc$get_queries_id_query()
- Orthanc$post_queries_id_retrieve()
- Orthanc$get_series()
- Orthanc$delete_series_id()
- Orthanc$get_series_id()
- Orthanc$post_series_id_anonymize()
- Orthanc$get_series_id_archive()
- Orthanc$post_series_id_archive()

- Orthanc$get_series_id_attachments()
- Orthanc$delete_series_id_attachments_name()
- Orthanc$get_series_id_attachments_name()
- Orthanc$put_series_id_attachments_name()
- Orthanc$post_series_id_attachments_name_compress()
- Orthanc$get_series_id_attachments_name_compressed_data()
- Orthanc$get_series_id_attachments_name_compressed_md5()
- Orthanc$get_series_id_attachments_name_compressed_size()
- Orthanc$get_series_id_attachments_name_data()
- Orthanc$get_series_id_attachments_name_info()
- Orthanc$get_series_id_attachments_name_is_compressed()
- Orthanc$get_series_id_attachments_name_md5()
- Orthanc$get_series_id_attachments_name_size()
- Orthanc$post_series_id_attachments_name_uncompress()
- Orthanc$post_series_id_attachments_name_verify_md5()
- Orthanc$get_series_id_instances()
- Orthanc$get_series_id_instances_tags()
- Orthanc$get_series_id_labels()
- Orthanc$delete_series_id_labels_label()
- Orthanc$get_series_id_labels_label()
- Orthanc$put_series_id_labels_label()
- Orthanc$get_series_id_media()
- Orthanc$post_series_id_media()
- Orthanc$get_series_id_metadata()
- Orthanc$delete_series_id_metadata_name()
- Orthanc$get_series_id_metadata_name()
- Orthanc$put_series_id_metadata_name()
- Orthanc$post_series_id_modify()
- Orthanc$get_series_id_module()
- Orthanc$get_series_id_numpy()
- Orthanc$get_series_id_patient()
- Orthanc$post_series_id_reconstruct()
- Orthanc$get_series_id_shared_tags()
- Orthanc$get_series_id_statistics()
- Orthanc$get_series_id_study()
- Orthanc$get_statistics()
- Orthanc$get_storage_commitment_id()
- Orthanc$post_storage_commitment_id_remove()
- Orthanc$get_studies()
- Orthanc$delete_studies_id()
- Orthanc$get_studies_id()
- Orthanc$post_studies_id_anonymize()

- Orthanc$get_studies_id_archive()
- Orthanc$post_studies_id_archive()
- Orthanc$get_studies_id_attachments()
- Orthanc$delete_studies_id_attachments_name()
- Orthanc$get_studies_id_attachments_name()
- Orthanc$put_studies_id_attachments_name()
- Orthanc$post_studies_id_attachments_name_compress()
- Orthanc$get_studies_id_attachments_name_compressed_data()
- Orthanc$get_studies_id_attachments_name_compressed_md5()
- Orthanc$get_studies_id_attachments_name_compressed_size()
- Orthanc$get_studies_id_attachments_name_data()
- Orthanc$get_studies_id_attachments_name_info()
- Orthanc$get_studies_id_attachments_name_is_compressed()
- Orthanc$get_studies_id_attachments_name_md5()
- Orthanc$get_studies_id_attachments_name_size()
- Orthanc$post_studies_id_attachments_name_uncompress()
- Orthanc$post_studies_id_attachments_name_verify_md5()
- Orthanc$get_studies_id_instances()
- Orthanc$get_studies_id_instances_tags()
- Orthanc$get_studies_id_labels()
- Orthanc$delete_studies_id_labels_label()
- Orthanc$get_studies_id_labels_label()
- Orthanc$put_studies_id_labels_label()
- Orthanc$get_studies_id_media()
- Orthanc$post_studies_id_media()
- Orthanc$post_studies_id_merge()
- Orthanc$get_studies_id_metadata()
- Orthanc$delete_studies_id_metadata_name()
- Orthanc$get_studies_id_metadata_name()
- Orthanc$put_studies_id_metadata_name()
- Orthanc$post_studies_id_modify()
- Orthanc$get_studies_id_module()
- Orthanc$get_studies_id_module_patient()
- Orthanc$get_studies_id_patient()
- Orthanc$post_studies_id_reconstruct()
- Orthanc$get_studies_id_series()
- Orthanc$get_studies_id_shared_tags()
- Orthanc$post_studies_id_split()
- Orthanc$get_studies_id_statistics()
- Orthanc$get_system()
- Orthanc$get_tools()
- Orthanc$get_tools_accepted_sop_classes()

- Orthanc$get_tools_accepted_transfer_syntaxes()
- Orthanc$put_tools_accepted_transfer_syntaxes()
- Orthanc$post_tools_bulk_anonymize()
- Orthanc$post_tools_bulk_content()
- Orthanc$post_tools_bulk_delete()
- Orthanc$post_tools_bulk_modify()
- Orthanc$post_tools_count_resources()
- Orthanc$get_tools_create_archive()
- Orthanc$post_tools_create_archive()
- Orthanc$post_tools_create_dicom()
- Orthanc$get_tools_create_media()
- Orthanc$post_tools_create_media()
- Orthanc$get_tools_create_media_extended()
- Orthanc$post_tools_create_media_extended()
- Orthanc$get_tools_default_encoding()
- Orthanc$put_tools_default_encoding()
- Orthanc$get_tools_dicom_conformance()
- Orthanc$post_tools_dicom_echo()
- Orthanc$post_tools_execute_script()
- Orthanc$post_tools_find()
- Orthanc$get_tools_generate_uid()
- Orthanc$post_tools_invalidate_tags()
- Orthanc$get_tools_labels()
- Orthanc$get_tools_log_level()
- Orthanc$put_tools_log_level()
- Orthanc$get_tools_log_level_dicom()
- Orthanc$put_tools_log_level_dicom()
- Orthanc$get_tools_log_level_generic()
- Orthanc$put_tools_log_level_generic()
- Orthanc$get_tools_log_level_http()
- Orthanc$put_tools_log_level_http()
- Orthanc$get_tools_log_level_jobs()
- Orthanc$put_tools_log_level_jobs()
- Orthanc$get_tools_log_level_lua()
- Orthanc$put_tools_log_level_lua()
- Orthanc$get_tools_log_level_plugins()
- Orthanc$put_tools_log_level_plugins()
- Orthanc$get_tools_log_level_sqlite()
- Orthanc$put_tools_log_level_sqlite()
- Orthanc$post_tools_lookup()
- Orthanc$get_tools_metrics()
- Orthanc$put_tools_metrics()

- Orthanc$get_tools_metrics_prometheus()
- Orthanc$get_tools_now()
- Orthanc$get_tools_now_local()
- Orthanc$post_tools_reconstruct()
- Orthanc$post_tools_reset()
- Orthanc$post_tools_shutdown()
- Orthanc$get_tools_unknown_sop_class_accepted()
- Orthanc$put_tools_unknown_sop_class_accepted()
- Orthanc$clone()

Method `new()`**:** Initialize a new Orthanc API Client

Usage:

```
Orthanc$new(
  url,
  username = NULL,
  password = NULL,
  params = NULL,
  headers = NULL,
  cookies = NULL,
  timeout = NULL,
  rate_limit = NULL,
  verify = NULL,
  ...
)
```

Arguments:

`url` URL for Orthanc REST API.

`username` Optional username for Basic HTTP authentication.

`password` Optional password for Basic HTTP authentication.

`params` Named-list of query parameters to include in every request.

`headers` Named-list of headers to include in every request.

`cookies` Named-list of cookies to set in every request.

`timeout` Maximum number of seconds to wait for a request to complete.

`rate_limit` Named-list with values `"capacity"` and `"fill_time"`. `"capacity"` is the maximum number of requests that can happen during the `"fill_time"` (in seconds). See `httr2::req_throttle()`.

`verify` Whether to verify the SSL/TLS certificate when making HTTP requests via `httr2`. Set to `FALSE` to disable certificate verification (e.g., when connecting to servers with self-signed certificates). Disabling verification reduces connection security and should only be used in trusted environments.

`...` Not currently used.

Method `GET()`**:** GET request with specified route

Usage:

```
Orthanc$GET(route, params = NULL, headers = NULL, cookies = NULL)
```

Arguments:

`route` HTTP route.

`params` Parameters for the HTTP request.

`headers` Headers for the HTTP request.

`cookies` Cookies for the HTTP request.

Returns: Serialized response of the HTTP GET request.

Method `DELETE()`**:** DELETE to specified route

Usage:

```
Orthanc$DELETE(route, params = NULL, headers = NULL, cookies = NULL)
```

Arguments:

`route` HTTP route.

`params` Parameters for the HTTP request.

`headers` Headers for the HTTP request.

`cookies` Cookies for the HTTP request.

Returns: Serialized response of the HTTP DELETE request.

Method `POST()`**:** POST to specified route

Usage:

```
Orthanc$POST(
  route,
  file = NULL,
  json = NULL,
  data = NULL,
  params = NULL,
  headers = NULL,
  cookies = NULL
)
```

Arguments:

`route` HTTP route.

`file` DICOM file to be uploaded or a ZIP archive containing DICOM files.

`json` List to be converted to JSON for request body.

`data` Raw data.

`params` Parameters for the HTTP request.

`headers` Headers for the HTTP request.

`cookies` Cookies for the HTTP request.

Returns: Serialized response of the HTTP POST request.

Method `PUT()`**:** PUT to specified route

Usage:

```
Orthanc$PUT(
  route,
  file = NULL,
  json = NULL,
```

```
  data = NULL,
  params = NULL,
  headers = NULL,
  cookies = NULL
)
```

Arguments:

`route` HTTP route.

`file` DICOM file to be uploaded or a ZIP archive containing DICOM files.

`json` List to be converted to JSON for request body.

`data` Raw data.

`params` Parameters for the HTTP request.

`headers` Headers for the HTTP request.

`cookies` Cookies for the HTTP request.

Returns: Serialized response of the HTTP PUT request.

Method `stream()`**:** Stream an HTTP response body and write to disk.

Usage:

```
Orthanc$stream(method, route, params = NULL, headers = NULL, cookies = NULL)
```

Arguments:

`method` HTTP method.

`route` HTTP route.

`params` Parameters for the HTTP request.

`headers` Headers for the HTTP request.

`cookies` Cookies for the HTTP request.

Returns: Serialized response of the HTTP GET request.

Method `print()`**:** Print method for `Orthanc`.

Usage:

```
Orthanc$print(x, ...)
```

Arguments:

`x` Object to print.

`...` Further arguments passed to or from other methods.

Method `delete_changes()`**:** Clear changes

Clear the full history stored in the changes log

Usage:

```
Orthanc$delete_changes()
```

Returns: Nothing, invisibly.

Method `get_changes()`**:** List changes

Whenever Orthanc receives a new DICOM instance, this event is recorded in the so-called *Changes Log*. This enables remote scripts to react to the arrival of new DICOM resources. A typical application is auto-routing, where an external script waits for a new DICOM instance to arrive into Orthanc, then forward this instance to another modality. Please note that, when resources are deleted, their corresponding change entries are also removed from the Changes Log, which helps ensuring that this log does not grow indefinitely.

Usage:

```
Orthanc$get_changes(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- last (number): Request only the last change id (this argument must be used alone)
- limit (number): Limit the number of results
- since (number): Show only the resources since the provided index excluded
- to (number): Show only the resources till the provided index included (only available if your DB backend supports ExtendedChanges)
- type (string): Show only the changes of the provided type (only available if your DB backend supports ExtendedChanges). Multiple values can be provided and must be separated by a ';'.

Returns: The list of changes.

Method `delete_exports()`**:** Clear exports

Clear the full history stored in the exports log

Usage:

```
Orthanc$delete_exports()
```

Returns: Nothing, invisibly.

Method `get_exports()`**:** List exports

For medical traceability, Orthanc can be configured to store a log of all the resources that have been exported to remote modalities. In auto-routing scenarios, it is important to prevent this log to grow indefinitely as incoming instances are routed. You can either disable this logging by setting the option `LogExportedResources` to `FALSE` in the configuration file, or periodically clear this log by `DELETE`-ing this URI. This route might be removed in future versions of Orthanc.

Usage:

```
Orthanc$get_exports(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- limit (number): Limit the number of results
- since (number): Show only the resources since the provided index

Returns: The list of exports.

Method `get_instances()`**:** List the available instances

List the Orthanc identifiers of all the available DICOM instances

Usage:

```
Orthanc$get_instances(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual resources, not only their Orthanc identifiers
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- limit (number): Limit the number of results
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- response-content (string): Defines the content of response for each returned resource. Allowed values are `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`, `Attachments`. If not specified, Orthanc will return `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`.e.g: 'response-content=MainDicomTags;Children (new in Orthanc 1.12.5 - overrides `expand`)
- short (boolean): If present, report the DICOM tags in hexadecimal format
- since (number): Show only the resources since the provided index

Returns: List containing either the Orthanc identifiers, or detailed information about the reported instances (if `expand` argument is provided).

Method `post_instances()`**:** Upload DICOM instances

Upload DICOM instances

Usage:

```
Orthanc$post_instances(file = NULL)
```

Arguments:

`file` (character) Path to file for request body. See Details.

Details: Request body: DICOM file to be uploaded (application/dicom) or ZIP archive containing DICOM files (new in Orthanc 1.8.2) (application/zip).

Returns: Information about the uploaded instance, or list of information for each uploaded instance in the case of ZIP archive.

Method `delete_instances_id()`**:** Delete some instance

Delete the DICOM instance whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$delete_instances_id(id)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

Returns: Nothing, invisibly.

Method `get_instances_id()`**:** Get information about some instance

Get detailed information about the DICOM instance whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_instances_id(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the DICOM instance.

Method `post_instances_id_anonymize()`**:** Anonymize instance

Download an anonymized version of the DICOM instance whose Orthanc identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/anonymization.html#anonymization-of-a-single-instance

Usage:

```
Orthanc$post_instances_id_anonymize(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- DicomVersion (character): Version of the DICOM standard to be used for anonymization. Check out configuration option `DeidentifyLogsDicomVersion` for possible values.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): List of DICOM tags whose value must not be destroyed by the anonymization. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- KeepLabels (logical): Keep the labels of all resources level (defaults to `FALSE`)
- KeepPrivateTags (logical): Keep the private tags from the DICOM instances (defaults to `FALSE`)
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- PrivateCreator (character): The private creator to be used for private tags in `Replace`

- Remove (list): List of additional tags to be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: The anonymized DICOM instance.

Method `get_instances_id_attachments()`**:** List attachments

Get the list of attachments that are associated with the given instance

Usage:

```
Orthanc$get_instances_id_attachments(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (string): If present, retrieve the attachments list and their numerical ids

Returns: List containing the names of the attachments.

Method `delete_instances_id_attachments_name()`**:** Delete attachment

Delete an attachment associated with the given DICOM instance. This call will fail if trying to delete a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_instances_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the attachment, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_instances_id_attachments_name()`**:** List operations on attachments

Get the list of the operations that are available for attachments associated with the given instance

Usage:

```
Orthanc$get_instances_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: List of the available operations.

Method `put_instances_id_attachments_name()`**:** Set attachment

Attach a file to the given DICOM instance. This call will fail if trying to modify a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_instances_id_attachments_name(
  id,
  name,
  headers = NULL,
  data = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Binary data containing the attachment (application/octet-stream). Optional headers (`headers`):

- If-Match (string): Revision of the attachment, if this is not the first time this attachment is set.

Returns: Empty JSON object in the case of a success.

Method `post_instances_id_attachments_name_compress()`**:** Compress attachment

Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_instances_id_attachments_name_compress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `get_instances_id_attachments_name_compressed_data()`**:** Get attachment (no decompression)

Get the (binary) content of one attachment associated with the given instance. The attachment will not be decompressed if `StorageCompression` is `TRUE`.

Usage:

```
Orthanc$get_instances_id_attachments_name_compressed_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_instances_id_attachments_name_compressed_md5()`**:** Get MD5 of attachment on disk

Get the MD5 hash of one attachment associated with the given instance, as stored on the disk. This is different from `.../md5` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_instances_id_attachments_name_compressed_md5(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment, as stored on the disk.

Method `get_instances_id_attachments_name_compressed_size()`**:** Get size of attachment on disk

Get the size of one attachment associated with the given instance, as stored on the disk. This is different from `.../size` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_instances_id_attachments_name_compressed_size(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment, as stored on the disk.

Method `get_instances_id_attachments_name_data()`**:** Get attachment

Get the (binary) content of one attachment associated with the given instance

Usage:

```
Orthanc$get_instances_id_attachments_name_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_instances_id_attachments_name_info()`**:** Get info about the attachment
Get all the information about the attachment associated with the given instance

Usage:

```
Orthanc$get_instances_id_attachments_name_info(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: JSON object containing the information about the attachment.

Method `get_instances_id_attachments_name_is_compressed()`**:** Is attachment compressed?
Test whether the attachment has been stored as a compressed file on the disk.

Usage:

```
Orthanc$get_instances_id_attachments_name_is_compressed(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: `0` if the attachment was stored uncompressed, `1` if it was compressed.

Method `get_instances_id_attachments_name_md5()`**:** Get MD5 of attachment
Get the MD5 hash of one attachment associated with the given instance

Usage:

```
Orthanc$get_instances_id_attachments_name_md5(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment.

Method `get_instances_id_attachments_name_size()`**:** Get size of attachment

Get the size of one attachment associated with the given instance

Usage:

```
Orthanc$get_instances_id_attachments_name_size(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment.

Method `post_instances_id_attachments_name_uncompress()`**:** Uncompress attachment

Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_instances_id_attachments_name_uncompress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `post_instances_id_attachments_name_verify_md5()`**:** Verify attachment

Verify that the attachment is not corrupted, by validating its MD5 hash

Usage:

```
Orthanc$post_instances_id_attachments_name_verify_md5(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: On success, a valid JSON object is returned.

Method `get_instances_id_content_path()`**:** Get raw tag

Get the raw content of one DICOM tag in the hierarchy of DICOM dataset

Usage:

```
Orthanc$get_instances_id_content_path(id, path)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`path` (character) Path to the DICOM tag. This is the interleaving of one DICOM tag, possibly followed by an index for sequences. Sequences are accessible as, for instance, `/0008-1140/1/0008-1150`.

Returns: The raw value of the tag of intereset (binary data, whose memory layout depends on the underlying transfer syntax), or List containing the list of available tags if accessing a dataset.

Method `post_instances_id_export()`**:** Write DICOM onto filesystem

Write the DICOM file onto the filesystem where Orthanc is running. This is insecure for Orthanc servers that are remotely accessible since one could overwrite any system file. Since Orthanc 1.12.0, this route is disabled by default, but can be enabled using the `RestApiWriteToFileSystemEnabled` configuration option.

Usage:

```
Orthanc$post_instances_id_export(id, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Target path on the filesystem (text/plain).

Returns: Nothing, invisibly.

Method `get_instances_id_file()`**:** Download DICOM

Download one DICOM instance

Usage:

```
Orthanc$get_instances_id_file(id, params = NULL, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): This HTTP header can be set to retrieve the DICOM instance in DICOMweb format

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM file will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: The DICOM instance.

Method `get_instances_id_frames()`**:** List available frames

List the frames that are available in the DICOM instance of interest

Usage:

```
Orthanc$get_instances_id_frames(id)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

Returns: The list of the indices of the available frames.

Method `get_instances_id_frames_frame()`**:** List operations

List the available operations under URI `/instances/{id}/frames/{frame}/`

Usage:

```
Orthanc$get_instances_id_frames_frame(id, frame)
```

Arguments:

`id` (character) .

`frame` (character) .

Returns: List of the available operations.

Method `get_instances_id_frames_frame_image_int16()`**:** Decode a frame (int16)

Decode one frame of interest from the given DICOM instance. Pixels of grayscale images are truncated to the [-32768,32767] range. Negative values must be interpreted according to two's complement.

Usage:

```
Orthanc$get_instances_id_frames_frame_image_int16(
  id,
  frame,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_frames_frame_image_uint16()`**:** Decode a frame (uint16)

Decode one frame of interest from the given DICOM instance. Pixels of grayscale images are truncated to the [0,65535] range.

Usage:

```
Orthanc$get_instances_id_frames_frame_image_uint16(
  id,
  frame,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_frames_frame_image_uint8()`**:** Decode a frame (uint8)

Decode one frame of interest from the given DICOM instance. Pixels of grayscale images are truncated to the [0,255] range.

Usage:

```
Orthanc$get_instances_id_frames_frame_image_uint8(
  id,
  frame,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_frames_frame_matlab()`**:** Decode frame for Matlab

Decode one frame of interest from the given DICOM instance, and export this frame as a Octave/Matlab matrix to be imported with `eval()`: https://orthanc.uclouvain.be/book/faq/matlab.html

Usage:

```
Orthanc$get_instances_id_frames_frame_matlab(id, frame)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

Returns: Octave/Matlab matrix.

Method `get_instances_id_frames_frame_numpy()`**:** Decode frame for numpy

Decode one frame of interest from the given DICOM instance, for use with numpy in Python. The numpy array has 3 dimensions: (height, width, color channel).

Usage:

```
Orthanc$get_instances_id_frames_frame_numpy(id, frame, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM resource of interest.

`frame` (numeric) Index of the frame (starts at `0`).

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- compress (boolean): Compress the file as `.npz`
- rescale (boolean): On grayscale images, apply the rescaling and return floating-point values

Returns: Numpy file: https://numpy.org/devdocs/reference/generated/numpy.lib.format.html.

Method `get_instances_id_frames_frame_preview()`**:** Decode a frame (preview)

Decode one frame of interest from the given DICOM instance. The full dynamic range of grayscale images is rescaled to the [0,255] range.

Usage:

```
Orthanc$get_instances_id_frames_frame_preview(
  id,
  frame,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_frames_frame_raw()`**:** Access raw frame

Access the raw content of one individual frame of the DICOM instance of interest, bypassing image decoding. This is notably useful to access the source files in compressed transfer syntaxes.

Usage:

```
Orthanc$get_instances_id_frames_frame_raw(id, frame)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

Returns: The raw frame.

Method `get_instances_id_frames_frame_raw.gz()`**:** Access raw frame (compressed)

Access the raw content of one individual frame of the DICOM instance of interest, bypassing image decoding. This is notably useful to access the source files in compressed transfer syntaxes. The image is compressed using gzip

Usage:

```
Orthanc$get_instances_id_frames_frame_raw.gz(id, frame)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

Returns: The raw frame, compressed using gzip.

Method `get_instances_id_frames_frame_rendered()`**:** Render a frame

Render one frame of interest from the given DICOM instance. This function takes scaling into account (`RescaleSlope` and `RescaleIntercept` tags), as well as the default windowing stored in the DICOM file (`WindowCenter` and `WindowWidthtags`), and can be used to resize the resulting image. Color images are not affected by windowing.

Usage:

```
Orthanc$get_instances_id_frames_frame_rendered(
  id,
  frame,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`frame` (numeric) Index of the frame (starts at `0`).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- height (number): Height of the resized image
- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)
- smooth (boolean): Whether to smooth image on resize
- width (number): Width of the resized image
- window-center (number): Windowing center
- window-width (number): Windowing width

Returns: JPEG image.

Method `get_instances_id_header()`**:** Get DICOM meta-header

Get the DICOM tags in the meta-header of the DICOM instance. By default, the `full` format is used, which combines hexadecimal tags with human-readable description.

Usage:

```
Orthanc$get_instances_id_header(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object containing the DICOM tags and their associated value.

Method `get_instances_id_image_int16()`**:** Decode an image (int16)

Decode the first frame of the given DICOM instance. Pixels of grayscale images are truncated to the [-32768,32767] range. Negative values must be interpreted according to two's complement.

Usage:

```
Orthanc$get_instances_id_image_int16(id, params = NULL, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_image_uint16()`**:** Decode an image (uint16)

Decode the first frame of the given DICOM instance. Pixels of grayscale images are truncated to the [0,65535] range.

Usage:

```
Orthanc$get_instances_id_image_uint16(id, params = NULL, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_image_uint8()`**:** Decode an image (uint8)

Decode the first frame of the given DICOM instance. Pixels of grayscale images are truncated to the [0,255] range.

Usage:

```
Orthanc$get_instances_id_image_uint8(id, params = NULL, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `get_instances_id_labels()`**:** List labels

Get the labels that are associated with the given instance (new in Orthanc 1.12.0)

Usage:

```
Orthanc$get_instances_id_labels(id)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

Returns: List containing the names of the labels.

Method `delete_instances_id_labels_label()`**:** Remove label

Remove a label associated with a instance

Usage:

```
Orthanc$delete_instances_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`label` (character) The label to be removed.

Returns: Nothing, invisibly.

Method `get_instances_id_labels_label()`**:** Test label

Test whether the instance is associated with the given label

Usage:

```
Orthanc$get_instances_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`label` (character) The label of interest.

Returns: Empty string is returned in the case of presence, error 404 in the case of absence.

Method `put_instances_id_labels_label()`**:** Add label

Associate a label with a instance

Usage:

```
Orthanc$put_instances_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`label` (character) The label to be added.

Returns: Nothing, invisibly.

Method `get_instances_id_matlab()`**:** Decode frame for Matlab

Decode the first frame of the given DICOM instance., and export this frame as a Octave/Matlab matrix to be imported with `eval()`: https://orthanc.uclouvain.be/book/faq/matlab.html

Usage:

```
Orthanc$get_instances_id_matlab(id)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

Returns: Octave/Matlab matrix.

Method `get_instances_id_metadata()`**:** List metadata

Get the list of metadata that are associated with the given instance

Usage:

```
Orthanc$get_instances_id_metadata(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, also retrieve the value of the individual metadata
- numeric (string): If present, use the numeric identifier of the metadata instead of its symbolic name

Returns: List containing the names of the available metadata, or List mapping metadata to their values (if `expand` argument is provided).

Method `delete_instances_id_metadata_name()`**:** Delete metadata

Delete some metadata associated with the given DICOM instance. This call will fail if trying to delete a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_instances_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the metadata, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_instances_id_metadata_name()`**:** Get metadata

Get the value of a metadata that is associated with the given instance

Usage:

```
Orthanc$get_instances_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the metadata, to check if its content has changed

Returns: Value of the metadata.

Method `put_instances_id_metadata_name()`**:** Set metadata

Set the value of some metadata in the given DICOM instance. This call will fail if trying to modify a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_instances_id_metadata_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: String value of the metadata (text/plain).
Optional headers (`headers`):

- If-Match (string): Revision of the metadata, if this is not the first time this metadata is set.

Returns: Nothing, invisibly.

Method `post_instances_id_modify()`**:** Modify instance

Download a modified version of the DICOM instance whose Orthanc identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/anonymization.html#modification-of-a-single-instance

Usage:

```
Orthanc$post_instances_id_modify(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): Keep the original value of the specified tags, to be chosen among the `StudyInstanceUID`, `SeriesInstanceUID` and `SOPInstanceUID` tags. Avoid this feature as much as possible, as this breaks the DICOM model of the real world.
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.

- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of tags that must be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- RemovePrivateTags (logical): Remove the private tags from the DICOM instances (defaults to `FALSE`)
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: The modified DICOM instance.

Method `get_instances_id_module()`**:** Get instance module

Get the instance module of the DICOM instance whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_instances_id_module(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: Information about the DICOM instance.

Method `get_instances_id_numpy()`**:** Decode instance for numpy

Decode the given DICOM instance, for use with numpy in Python. The numpy array has 4 dimensions: (frame, height, width, color channel).

Usage:

```
Orthanc$get_instances_id_numpy(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM resource of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- compress (boolean): Compress the file as `.npz`
- rescale (boolean): On grayscale images, apply the rescaling and return floating-point values

Returns: Numpy file: https://numpy.org/devdocs/reference/generated/numpy.lib.format.html.

Method `get_instances_id_patient()`**:** Get parent patient

Get detailed information about the parent patient of the DICOM instance whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_instances_id_patient(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the parent DICOM patient.

Method `get_instances_id_pdf()`**:** Get embedded PDF

Get the PDF file that is embedded in one DICOM instance. If the DICOM instance doesn't contain the `EncapsulatedDocument` tag or if the `MIMETypeOfEncapsulatedDocument` tag doesn't correspond to the PDF type, a `404` HTTP error is raised.

Usage:

```
Orthanc$get_instances_id_pdf(id)
```

Arguments:

`id` (character) Orthanc identifier of the instance interest.

Returns: PDF file.

Method `get_instances_id_preview()`**:** Decode an image (preview)

Decode the first frame of the given DICOM instance. The full dynamic range of grayscale images is rescaled to the [0,255] range.

Usage:

```
Orthanc$get_instances_id_preview(id, params = NULL, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)

Returns: JPEG image.

Method `post_instances_id_reconstruct()`**:** Reconstruct tags & optionally files of instance

Reconstruct the main DICOM tags in DB of the instance whose Orthanc identifier is provided in the URL. This is useful if child studies/series/instances have inconsistent values for higher-level tags, in order to force Orthanc to use the value from the resource of interest. Beware that this is a time-consuming operation, as all the children DICOM instances will be parsed again, and the Orthanc index will be updated accordingly.

Usage:

```
Orthanc$post_instances_id_reconstruct(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- LimitToThisLevelMainDicomTags (logical): Only reconstruct this level MainDicomTags by re-reading them from a random child instance of the resource. This option is much faster than a full reconstruct and is useful e.g. if you have modified the 'ExtraMainDicomTags' at the Study level to optimize the speed of some C-Find. 'false' by default. (New in Orthanc 1.12.4)
- ReconstructFiles (logical): Also reconstruct the files of the resources (e.g: apply IngestTranscoding, StorageCompression). 'false' by default. (New in Orthanc 1.11.0)

Returns: Nothing, invisibly.

Method `get_instances_id_rendered()`**:** Render an image

Render the first frame of the given DICOM instance. This function takes scaling into account (`RescaleSlope` and `RescaleIntercept` tags), as well as the default windowing stored in the DICOM file (`WindowCenter` and `WindowWidth`tags), and can be used to resize the resulting image. Color images are not affected by windowing.

Usage:

```
Orthanc$get_instances_id_rendered(id, params = NULL, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Accept (string): Format of the resulting image. Can be `image/png` (default), `image/jpeg` or `image/x-portable-arbitrarymap`

Optional query parameters (`params`):

- height (number): Height of the resized image
- quality (number): Quality for JPEG images (between 1 and 100, defaults to 90)
- returnUnsupportedImage (boolean): Returns an unsupported.png placeholder image if unable to provide the image instead of returning a 415 HTTP error (value is true if option is present)
- smooth (boolean): Whether to smooth image on resize
- width (number): Width of the resized image
- window-center (number): Windowing center
- window-width (number): Windowing width

Returns: JPEG image.

Method `get_instances_id_series()`**:** Get parent series

Get detailed information about the parent series of the DICOM instance whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_instances_id_series(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the parent DICOM series.

Method `get_instances_id_simplified_tags()`**:** Get human-readable tags

Get the DICOM tags in human-readable format (same as the `/instances/{id}/tags?simplify` route)

Usage:

```
Orthanc$get_instances_id_simplified_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long

- whole (boolean): Whether to read the whole DICOM file from the storage area (new in Orthanc 1.12.4). If set to "false" (default value), the DICOM file is read until the pixel data tag (7fe0,0010) to optimize access to storage. Setting the option to "true" provides access to the DICOM tags stored after the pixel data tag.

Returns: JSON object containing the DICOM tags and their associated value.

Method `get_instances_id_statistics()`**:** Get instance statistics

Get statistics about the given instance

Usage:

```
Orthanc$get_instances_id_statistics(id)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

Returns: Nothing, invisibly.

Method `get_instances_id_study()`**:** Get parent study

Get detailed information about the parent study of the DICOM instance whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_instances_id_study(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the parent DICOM study.

Method `get_instances_id_tags()`**:** Get DICOM tags

Get the DICOM tags in the specified format. By default, the `full` format is used, which combines hexadecimal tags with human-readable description.

Usage:

```
Orthanc$get_instances_id_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM instance of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)
- whole (boolean): Whether to read the whole DICOM file from the storage area (new in Orthanc 1.12.4). If set to "false" (default value), the DICOM file is read until the pixel data tag (7fe0,0010) to optimize access to storage. Setting the option to "true" provides access to the DICOM tags stored after the pixel data tag.

Returns: JSON object containing the DICOM tags and their associated value.

Method `get_jobs()`**:** List jobs

List all the available jobs

Usage:

```
Orthanc$get_jobs(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual jobs

Returns: List containing either the jobs identifiers, or detailed information about the reported jobs (if `expand` argument is provided).

Method `delete_jobs_id()`**:** Delete a job from history

Delete the job from the jobs history. Only a completed job can be deleted. If the job has not run or not completed yet, you must cancel it first. If the job has outputs, all outputs will be deleted as well.

Usage:

```
Orthanc$delete_jobs_id(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: Nothing, invisibly.

Method `get_jobs_id()`**:** Get job

Retrieve detailed information about the job whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users rest.html#jobs

Usage:

```
Orthanc$get_jobs_id(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: JSON object detailing the job.

Method `post_jobs_id_cancel()`**:** Cancel job

Cancel the job whose identifier is provided in the URL. Check out the Orthanc Book for more information about the state machine applicable to jobs: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs

Usage:

```
Orthanc$post_jobs_id_cancel(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: Empty JSON object in the case of a success.

Method `post_jobs_id_pause()`**:** Pause job

Pause the job whose identifier is provided in the URL. Check out the Orthanc Book for more information about the state machine applicable to jobs: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs

Usage:

```
Orthanc$post_jobs_id_pause(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: Empty JSON object in the case of a success.

Method `post_jobs_id_resubmit()`**:** Resubmit job

Resubmit the job whose identifier is provided in the URL. Check out the Orthanc Book for more information about the state machine applicable to jobs: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs

Usage:

```
Orthanc$post_jobs_id_resubmit(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: Empty JSON object in the case of a success.

Method `post_jobs_id_resume()`**:** Resume job

Resume the job whose identifier is provided in the URL. Check out the Orthanc Book for more information about the state machine applicable to jobs: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs

Usage:

```
Orthanc$post_jobs_id_resume(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: Empty JSON object in the case of a success.

Method `delete_jobs_id_key()`**:** Delete a job output

Delete the output produced by a job. As of Orthanc 1.12.1, only the jobs that generate a DICOMDIR media or a ZIP archive provide such an output (with `key` equals to `archive`).

Usage:

```
Orthanc$delete_jobs_id_key(id, key)
```

Arguments:

`id` (character) Identifier of the job of interest.

`key` (character) Name of the output of interest.

Returns: Nothing, invisibly.

Method `get_jobs_id_key()`**:** Get job output

Retrieve some output produced by a job. As of Orthanc 1.8.2, only the jobs that generate a DICOMDIR media or a ZIP archive provide such an output (with `key` equals to `archive`).

Usage:

```
Orthanc$get_jobs_id_key(id, key)
```

Arguments:

`id` (character) Identifier of the job of interest.

`key` (character) Name of the output of interest.

Returns: Content of the output of the job.

Method `get_modalities()`**:** List DICOM modalities

List all the DICOM modalities that are known to Orthanc. This corresponds either to the content of the `DicomModalities` configuration option, or to the information stored in the database if `DicomModalitiesInDatabase` is `TRUE`.

Usage:

```
Orthanc$get_modalities(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual DICOM modalities

Returns: List containing either the identifiers of the modalities, or detailed information about the modalities (if `expand` argument is provided).

Method `delete_modalities_id()`**:** Delete DICOM modality

Delete one DICOM modality. This change is permanent iff. `DicomModalitiesInDatabase` is `TRUE`, otherwise it is lost at the next restart of Orthanc.

Usage:

```
Orthanc$delete_modalities_id(id)
```

Arguments:

`id` (character) Identifier of the DICOM modality of interest.

Returns: Nothing, invisibly.

Method `get_modalities_id()`**:** List operations on modality

List the operations that are available for a DICOM modality.

Usage:

```
Orthanc$get_modalities_id(id)
```

Arguments:

`id` (character) Identifier of the DICOM modality of interest.

Returns: List of the available operations.

Method `put_modalities_id()`**:** Update DICOM modality

Define a new DICOM modality, or update an existing one. This change is permanent iff. `DicomModalitiesInDatabase` is `TRUE`, otherwise it is lost at the next restart of Orthanc.

Usage:

```
Orthanc$put_modalities_id(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the new/updated DICOM modality.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- AET (character): AET of the remote DICOM modality
- AllowEcho (logical): Whether to accept C-ECHO SCU commands issued by the remote modality
- AllowFind (logical): Whether to accept C-FIND SCU commands issued by the remote modality
- AllowFindWorklist (logical): Whether to accept C-FIND SCU commands for worklists issued by the remote modality
- AllowGet (logical): Whether to accept C-GET SCU commands issued by the remote modality
- AllowMove (logical): Whether to accept C-MOVE SCU commands issued by the remote modality
- AllowStorageCommitment (logical): Whether to accept storage commitment requests issued by the remote modality
- AllowStore (logical): Whether to accept C-STORE SCU commands issued by the remote modality
- AllowTranscoding (logical): Whether to allow transcoding for operations initiated by this modality. This option applies to Orthanc C-GET SCP and to Orthanc C-STORE SCU. It only has an effect if the global option `EnableTranscoding` is set to `TRUE`.
- Host (character): Host address of the remote DICOM modality (typically, an IP address)
- LocalAet (character): Whether to override the default DicomAet in the SCU connection initiated by Orthanc to this modality
- Manufacturer (character): Manufacturer of the remote DICOM modality (check configuration option `DicomModalities` for possible values
- Port (numeric): TCP port of the remote DICOM modality
- Timeout (numeric): Whether to override the default DicomScuTimeout in the SCU connection initiated by Orthanc to this modality
- UseDicomTls (logical): Whether to use DICOM TLS in the SCU connection initiated by Orthanc (new in Orthanc 1.9.0)

Returns: Nothing, invisibly.

Method `get_modalities_id_configuration()`**:** Get modality configuration

Get detailed information about the configuration of some DICOM modality

Usage:

```
Orthanc$get_modalities_id_configuration(id)
```

Arguments:

`id` (character) Identifier of the modality of interest.

Returns: Configuration of the modality.

Method `post_modalities_id_echo()`**:** Trigger C-ECHO SCU

Trigger C-ECHO SCU command against the DICOM modality whose identifier is provided in URL: https://orthanc.uclouvain.be/book/users/rest.html#performing-c-echo

Usage:

```
Orthanc$post_modalities_id_echo(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- CheckFind (logical): Issue a dummy C-FIND command after the C-GET SCU, in order to check whether the remote modality knows about Orthanc. This field defaults to the value of the `DicomEchoChecksFind` configuration option. New in Orthanc 1.8.1.
- Timeout (numeric): Timeout for the C-ECHO command, in seconds

Returns: Nothing, invisibly.

Method `post_modalities_id_find_worklist()`**:** C-FIND SCU for worklist

Trigger C-FIND SCU command against the remote worklists of the DICOM modality whose identifier is provided in URL

Usage:

```
Orthanc$post_modalities_id_find_worklist(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Full (logical): If set to `TRUE`, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- Query (list): Associative array containing the filter on the values of the DICOM tags
- Short (logical): If set to `TRUE`, report the DICOM tags in hexadecimal format

Returns: List describing the DICOM tags of the matching worklists.

Method `post_modalities_id_get()`**:** Trigger C-GET SCU

Start a C-GET SCU command as a job, in order to retrieve DICOM resources from a remote DICOM modality whose identifier is provided in the URL:

Usage:

```
Orthanc$post_modalities_id_get(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Level (character): Level of the query (`Patient`, `Study`, `Series` or `Instance`)
- LocalAet (character): Local AET that is used for this commands, defaults to `DicomAet` configuration option. Ignored if `DicomModalities` already sets `LocalAet` for this modality.
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- Resources (list): List of queries identifying all the DICOM resources to be sent. Usage of wildcards is prohibited and the query shall only contain DICOM ID tags. Additionally, you may provide SOPClassesInStudy to limit the scope of the DICOM negotiation to certain SOPClassUID or to present uncommon SOPClassUID during the DICOM negotiation. By default, Orhanc will propose the most 120 common SOPClassUIDs.
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Timeout (numeric): Timeout for the C-GET command, in seconds
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `post_modalities_id_move()`**:** Trigger C-MOVE SCU

Start a C-MOVE SCU command as a job, in order to drive the execution of a sequence of C-STORE commands by some remote DICOM modality whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/rest.html#performing-c-move

Usage:

```
Orthanc$post_modalities_id_move(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Level (character): Level of the query (`Patient`, `Study`, `Series` or `Instance`)
- LocalAet (character): Local AET that is used for this commands, defaults to `DicomAet` configuration option. Ignored if `DicomModalities` already sets `LocalAet` for this modality.

- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- Resources (list): List of queries identifying all the DICOM resources to be sent
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- TargetAet (character): Target AET that will be used by the remote DICOM modality as a target for its C-STORE SCU commands, defaults to `DicomAet` configuration option in order to do a simple query/retrieve
- Timeout (numeric): Timeout for the C-MOVE command, in seconds
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `post_modalities_id_query()`**:** Trigger C-FIND SCU

Trigger C-FIND SCU command against the DICOM modality whose identifier is provided in URL: https://orthanc.uclouvain.be/book/users/rest.html#performing-query-retrieve-c-find-and-find-with-rest

Usage:

```
Orthanc$post_modalities_id_query(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Level (character): Level of the query (`Patient`, `Study`, `Series` or `Instance`)
- LocalAet (character): Local AET that is used for this commands, defaults to `DicomAet` configuration option. Ignored if `DicomModalities` already sets `LocalAet` for this modality.
- Normalize (logical): Whether to normalize the query, i.e. whether to wipe out from the query, the DICOM tags that are not applicable for the query-retrieve level of interest
- Query (list): Associative array containing the filter on the values of the DICOM tags
- Timeout (numeric): Timeout for the C-FIND command and subsequent C-MOVE retrievals, in seconds (new in Orthanc 1.9.1)

Returns: Nothing, invisibly.

Method `post_modalities_id_storage_commitment()`**:** Trigger storage commitment request

Trigger a storage commitment request to some remote DICOM modality whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/storage-commitment.html#storage-commitment-scu

Usage:

```
Orthanc$post_modalities_id_storage_commitment(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- DicomInstances (list): List of DICOM resources that are not necessarily stored within Orthanc, but that must be checked by storage commitment. This is a list of JSON objects that must contain the `SOPClassUID` and `SOPInstanceUID` fields.
- Resources (list): List of the Orthanc identifiers of the DICOM resources to be checked by storage commitment
- Timeout (numeric): Timeout for the storage commitment command (new in Orthanc 1.9.1)

Returns: Nothing, invisibly.

Method `post_modalities_id_store()`**:** Trigger C-STORE SCU

Start a C-STORE SCU command as a job, in order to send DICOM resources stored locally to some remote DICOM modality whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/rest.html#r store-scu

Usage:

```
Orthanc$post_modalities_id_store(id, json = NULL, data = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- CalledAet (character): Called AET that is used for this commands, defaults to `AET` configuration option. Allows you to overwrite the destination AET for a specific operation.
- Host (character): Host that is used for this commands, defaults to `Host` configuration option. Allows you to overwrite the destination host for a specific operation.
- LocalAet (character): Local AET that is used for this commands, defaults to `DicomAet` configuration option. Ignored if `DicomModalities` already sets `LocalAet` for this modality.
- MoveOriginatorAet (character): Move originator AET that is used for this commands, in order to fake a C-MOVE SCU
- MoveOriginatorID (numeric): Move originator ID that is used for this commands, in order to fake a C-MOVE SCU
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Port (numeric): Port that is used for this command, defaults to `Port` configuration option. Allows you to overwrite the destination port for a specific operation.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- Resources (list): List of the Orthanc identifiers of all the DICOM resources to be sent
- StorageCommitment (logical): Whether to chain C-STORE with DICOM storage commitment to validate the success of the transmission: https://orthanc.uclouvain.be/book/users/storage-commitment.html#chaining-c-store-with-storage-commitment

- Synchronous (logical): If TRUE, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Timeout (numeric): Timeout for the C-STORE command, in seconds
- UserData (list): User data that will travel along with the job. Request body: The Orthanc identifier of one resource to be sent (text/plain).

Returns: Nothing, invisibly.

Method `post_modalities_id_store_straight()`**:** Straight C-STORE SCU

Synchronously send the DICOM instance in the POST body to the remote DICOM modality whose identifier is provided in URL, without having to first store it locally within Orthanc. This is an alternative to command-line tools such as `storescu` from DCMTK or dcm4che.

Usage:

```
Orthanc$post_modalities_id_store_straight(id, file = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`file` (character) Path to file for request body. See Details.

Details: Request body: DICOM instance to be sent (application/dicom).

Returns: Nothing, invisibly.

Method `get_patients()`**:** List the available patients

List the Orthanc identifiers of all the available DICOM patients

Usage:

```
Orthanc$get_patients(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual resources, not only their Orthanc identifiers
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- limit (number): Limit the number of results
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- response-content (string): Defines the content of response for each returned resource. Allowed values are `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`, `Attachments`. If not specified, Orthanc will return `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`.e.g: 'response-content=MainDicomTags;Children (new in Orthanc 1.12.5 - overrides `expand`)

- short (boolean): If present, report the DICOM tags in hexadecimal format
- since (number): Show only the resources since the provided index

Returns: List containing either the Orthanc identifiers, or detailed information about the reported patients (if `expand` argument is provided).

Method `delete_patients_id()`**:** Delete some patient

Delete the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$delete_patients_id(id)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

Returns: Nothing, invisibly.

Method `get_patients_id()`**:** Get information about some patient

Get detailed information about the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the DICOM patient.

Method `post_patients_id_anonymize()`**:** Anonymize patient

Start a job that will anonymize all the DICOM instances within the patient whose identifier is provided in the URL. The modified DICOM instances will be stored into a brand new patient, whose Orthanc identifiers will be returned by the job. https://orthanc.uclouvain.be/book/users/anonymization.html#anonymization-of-patients-studies-or-series

Usage:

```
Orthanc$post_patients_id_anonymize(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- DicomVersion (character): Version of the DICOM standard to be used for anonymization. Check out configuration option `DeidentifyLogsDicomVersion` for possible values.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): List of DICOM tags whose value must not be destroyed by the anonymization. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- KeepLabels (logical): Keep the labels of all resources level (defaults to `FALSE`)
- KeepPrivateTags (logical): Keep the private tags from the DICOM instances (defaults to `FALSE`)
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of additional tags to be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_patients_id_archive()`**:** Create ZIP archive

Synchronously create a ZIP archive containing the DICOM patient whose Orthanc identifier is provided in the URL. This flavor is synchronous, which might *not* be desirable to archive large amount of data, as it might lead to network timeouts. Prefer the asynchronous version using `POST` method.

Usage:

```
Orthanc$get_patients_id_archive(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: ZIP file containing the archive.

Method `post_patients_id_archive()`**:** Create ZIP archive

Create a ZIP archive containing the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$post_patients_id_archive(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_patients_id_attachments()`**:** List attachments

Get the list of attachments that are associated with the given patient

Usage:

```
Orthanc$get_patients_id_attachments(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (string): If present, retrieve the attachments list and their numerical ids

Returns: List containing the names of the attachments.

Method `delete_patients_id_attachments_name()`**:** Delete attachment

Delete an attachment associated with the given DICOM patient. This call will fail if trying to delete a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_patients_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the attachment, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_patients_id_attachments_name()`**:** List operations on attachments

Get the list of the operations that are available for attachments associated with the given patient

Usage:

```
Orthanc$get_patients_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: List of the available operations.

Method `put_patients_id_attachments_name()`**:** Set attachment

Attach a file to the given DICOM patient. This call will fail if trying to modify a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_patients_id_attachments_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Binary data containing the attachment (application/octet-stream).
Optional headers (`headers`):

- If-Match (string): Revision of the attachment, if this is not the first time this attachment is set.

Returns: Empty JSON object in the case of a success.

Method `post_patients_id_attachments_name_compress()`**:** Compress attachment

Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_patients_id_attachments_name_compress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `get_patients_id_attachments_name_compressed_data()`**:** Get attachment (no decompression)

Get the (binary) content of one attachment associated with the given patient. The attachment will not be decompressed if `StorageCompression` is `TRUE`.

Usage:

```
Orthanc$get_patients_id_attachments_name_compressed_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_patients_id_attachments_name_compressed_md5()`**:** Get MD5 of attachment on disk

Get the MD5 hash of one attachment associated with the given patient, as stored on the disk. This is different from `.../md5` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_patients_id_attachments_name_compressed_md5(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment, as stored on the disk.

Method `get_patients_id_attachments_name_compressed_size()`**:** Get size of attachment on disk

Get the size of one attachment associated with the given patient, as stored on the disk. This is different from `.../size` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_patients_id_attachments_name_compressed_size(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment, as stored on the disk.

Method `get_patients_id_attachments_name_data()`**:** Get attachment

Get the (binary) content of one attachment associated with the given patient

Usage:

```
Orthanc$get_patients_id_attachments_name_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_patients_id_attachments_name_info()`**:** Get info about the attachment

Get all the information about the attachment associated with the given patient

Usage:

```
Orthanc$get_patients_id_attachments_name_info(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: JSON object containing the information about the attachment.

Method `get_patients_id_attachments_name_is_compressed()`**:** Is attachment compressed?
Test whether the attachment has been stored as a compressed file on the disk.

Usage:

```
Orthanc$get_patients_id_attachments_name_is_compressed(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: `0` if the attachment was stored uncompressed, `1` if it was compressed.

Method `get_patients_id_attachments_name_md5()`**:** Get MD5 of attachment
Get the MD5 hash of one attachment associated with the given patient

Usage:

```
Orthanc$get_patients_id_attachments_name_md5(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment.

Method `get_patients_id_attachments_name_size()`**:** Get size of attachment
Get the size of one attachment associated with the given patient

Usage:

```
Orthanc$get_patients_id_attachments_name_size(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment.

Method `post_patients_id_attachments_name_uncompress()`**:** Uncompress attachment
Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_patients_id_attachments_name_uncompress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `post_patients_id_attachments_name_verify_md5()`**:** Verify attachment
Verify that the attachment is not corrupted, by validating its MD5 hash

Usage:

```
Orthanc$post_patients_id_attachments_name_verify_md5(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: On success, a valid JSON object is returned.

Method `get_patients_id_instances()`**:** Get child instances
Get detailed information about the child instances of the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id_instances(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If false or missing, only retrieve the list of child instances
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.

- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: List containing information about the child DICOM instances.

Method `get_patients_id_instances_tags()`**:** Get tags of instances

Get the tags of all the child instances of the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id_instances_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object associating the Orthanc identifiers of the instances, with the values of their DICOM tags.

Method `get_patients_id_labels()`**:** List labels

Get the labels that are associated with the given patient (new in Orthanc 1.12.0)

Usage:

```
Orthanc$get_patients_id_labels(id)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

Returns: List containing the names of the labels.

Method `delete_patients_id_labels_label()`**:** Remove label

Remove a label associated with a patient

Usage:

```
Orthanc$delete_patients_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`label` (character) The label to be removed.

Returns: Nothing, invisibly.

Method `get_patients_id_labels_label()`**:** Test label

Test whether the patient is associated with the given label

Usage:

```
Orthanc$get_patients_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`label` (character) The label of interest.

Returns: Empty string is returned in the case of presence, error 404 in the case of absence.

Method `put_patients_id_labels_label()`**:** Add label

Associate a label with a patient

Usage:

```
Orthanc$put_patients_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`label` (character) The label to be added.

Returns: Nothing, invisibly.

Method `get_patients_id_media()`**:** Create DICOMDIR media

Synchronously create a DICOMDIR media containing the DICOM patient whose Orthanc identifier is provided in the URL. This flavor is synchronous, which might *not* be desirable to archive large amount of data, as it might lead to network timeouts. Prefer the asynchronous version using `POST` method.

Usage:

```
Orthanc$get_patients_id_media(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- extended (string): If present, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*
- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: ZIP file containing the archive.

Method `post_patients_id_media()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$post_patients_id_media(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Extended (logical): If `TRUE`, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*. Default value is `FALSE`.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_patients_id_metadata()`**:** List metadata

Get the list of metadata that are associated with the given patient

Usage:

```
Orthanc$get_patients_id_metadata(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, also retrieve the value of the individual metadata
- numeric (string): If present, use the numeric identifier of the metadata instead of its symbolic name

Returns: List containing the names of the available metadata, or List mapping metadata to their values (if `expand` argument is provided).

Method `delete_patients_id_metadata_name()`**:** Delete metadata

Delete some metadata associated with the given DICOM patient. This call will fail if trying to delete a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_patients_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the metadata, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_patients_id_metadata_name()`**:** Get metadata

Get the value of a metadata that is associated with the given patient

Usage:

```
Orthanc$get_patients_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the metadata, to check if its content has changed

Returns: Value of the metadata.

Method `put_patients_id_metadata_name()`**:** Set metadata

Set the value of some metadata in the given DICOM patient. This call will fail if trying to modify a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_patients_id_metadata_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: String value of the metadata (text/plain).
Optional headers (`headers`):

- If-Match (string): Revision of the metadata, if this is not the first time this metadata is set.

Returns: Nothing, invisibly.

Method `post_patients_id_modify()`**:** Modify patient

Start a job that will modify all the DICOM instances within the patient whose identifier is provided in the URL. The modified DICOM instances will be stored into a brand new patient, whose Orthanc identifiers will be returned by the job. https://orthanc.uclouvain.be/book/users/anonymization.html#modification-of-studies-or-series

Usage:

```
Orthanc$post_patients_id_modify(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): Keep the original value of the specified tags, to be chosen among the `StudyInstanceUID`, `SeriesInstanceUID` and `SOPInstanceUID` tags. Avoid this feature as much as possible, as this breaks the DICOM model of the real world.
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of tags that must be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- RemovePrivateTags (logical): Remove the private tags from the DICOM instances (defaults to `FALSE`)
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_patients_id_module()`**:** Get patient module

Get the patient module of the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id_module(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: Information about the DICOM patient.

Method `get_patients_id_protected()`**:** Is the patient protected against recycling?

Is the patient protected against recycling?

Usage:

```
Orthanc$get_patients_id_protected(id)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

Returns: `1` if protected, `0` if not protected.

Method `put_patients_id_protected()`**:** Protect/Unprotect a patient against recycling

Protects a patient by sending `1` or `TRUE` in the payload request. Unprotects a patient by sending `0` or `FALSE` in the payload requests. More info: https://orthanc.uclouvain.be/book/faq/features.html#recycling-protection

Usage:

```
Orthanc$put_patients_id_protected(id, json)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`json` (list) List for request body.

Returns: Nothing, invisibly.

Method `post_patients_id_reconstruct()`**:** Reconstruct tags & optionally files of patient

Reconstruct the main DICOM tags in DB of the patient whose Orthanc identifier is provided in the URL. This is useful if child studies/series/instances have inconsistent values for higher-level tags, in order to force Orthanc to use the value from the resource of interest. Beware that this is a time-consuming operation, as all the children DICOM instances will be parsed again, and the Orthanc index will be updated accordingly.

Usage:

```
Orthanc$post_patients_id_reconstruct(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- LimitToThisLevelMainDicomTags (logical): Only reconstruct this level MainDicomTags by re-reading them from a random child instance of the resource. This option is much faster than a full reconstruct and is useful e.g. if you have modified the 'ExtraMainDicomTags' at the Study level to optimize the speed of some C-Find. 'false' by default. (New in Orthanc 1.12.4)
- ReconstructFiles (logical): Also reconstruct the files of the resources (e.g: apply IngestTranscoding, StorageCompression). 'false' by default. (New in Orthanc 1.11.0)

Returns: Nothing, invisibly.

Method `get_patients_id_series()`**:** Get child series

Get detailed information about the child series of the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id_series(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If false or missing, only retrieve the list of child series
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'RequestedTags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: List containing information about the child DICOM series.

Method `get_patients_id_shared_tags()`**:** Get shared tags

Extract the DICOM tags whose value is constant across all the child instances of the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id_shared_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object containing the values of the DICOM tags.

Method `get_patients_id_statistics()`**:** Get patient statistics

Get statistics about the given patient

Usage:

```
Orthanc$get_patients_id_statistics(id)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

Returns: Nothing, invisibly.

Method `get_patients_id_studies()`**:** Get child studies

Get detailed information about the child studies of the DICOM patient whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_patients_id_studies(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the patient of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If false or missing, only retrieve the list of child studies
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: List containing information about the child DICOM studies.

Method `get_peers()`**:** List Orthanc peers

List all the Orthanc peers that are known to Orthanc. This corresponds either to the content of the `OrthancPeers` configuration option, or to the information stored in the database if `OrthancPeersInDatabase` is `TRUE`.

Usage:

```
Orthanc$get_peers(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual Orthanc peers

Returns: List containing either the identifiers of the peers, or detailed information about the peers (if `expand` argument is provided).

Method `delete_peers_id()`**:** Delete Orthanc peer

Delete one Orthanc peer. This change is permanent iff. `OrthancPeersInDatabase` is `TRUE`, otherwise it is lost at the next restart of Orthanc.

Usage:

```
Orthanc$delete_peers_id(id)
```

Arguments:

`id` (character) Identifier of the Orthanc peer of interest.

Returns: Nothing, invisibly.

Method `get_peers_id()`**:** List operations on peer

List the operations that are available for an Orthanc peer.

Usage:

```
Orthanc$get_peers_id(id)
```

Arguments:

`id` (character) Identifier of the peer of interest.

Returns: List of the available operations.

Method `put_peers_id()`**:** Update Orthanc peer

Define a new Orthanc peer, or update an existing one. This change is permanent iff. `OrthancPeersInDatabase` is `TRUE`, otherwise it is lost at the next restart of Orthanc.

Usage:

```
Orthanc$put_peers_id(id, json = NULL)
```

Arguments:

`id` (character) Identifier of the new/updated Orthanc peer.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- CertificateFile (character): SSL certificate for the HTTPS connections
- CertificateKeyFile (character): Key file for the SSL certificate for the HTTPS connections
- CertificateKeyPassword (character): Key password for the SSL certificate for the HTTPS connections
- HttpHeaders (list): HTTP headers to be used for the connections to the remote peer
- Password (character): Password for the credentials
- URL (character): URL of the root of the REST API of the remote Orthanc peer, for instance `http://localhost:8042/`
- Username (character): Username for the credentials

Returns: Nothing, invisibly.

Method `get_peers_id_configuration()`**:** Get peer configuration

Get detailed information about the configuration of some Orthanc peer

Usage:

```
Orthanc$get_peers_id_configuration(id)
```

Arguments:

`id` (character) Identifier of the peer of interest.

Returns: Configuration of the peer.

Method `post_peers_id_store()`**:** Send to Orthanc peer

Send DICOM resources stored locally to some remote Orthanc peer whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/rest.html#sending-one-resource

Usage:

```
Orthanc$post_peers_id_store(id, json = NULL, data = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`json` (list) Named-list for request body. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Compress (logical): Whether to compress the DICOM instances using gzip before the actual sending
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- Resources (list): List of the Orthanc identifiers of all the DICOM resources to be sent
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode to the provided DICOM transfer syntax before the actual sending
- UserData (list): User data that will travel along with the job. Request body: The Orthanc identifier of one resource to be sent (text/plain).

Returns: Nothing, invisibly.

Method `post_peers_id_store_straight()`**:** Straight store to peer

Synchronously send the DICOM instance in the POST body to the Orthanc peer whose identifier is provided in URL, without having to first store it locally within Orthanc. This is an alternative to command-line tools such as `curl`.

Usage:

```
Orthanc$post_peers_id_store_straight(id, file = NULL)
```

Arguments:

`id` (character) Identifier of the modality of interest.

`file` (character) Path to file for request body. See Details.

Details: Request body: DICOM instance to be sent (application/dicom).

Returns: Nothing, invisibly.

Method `get_peers_id_system()`**:** Get peer system information

Get system information about some Orthanc peer. This corresponds to doing a `GET` request against the `/system` URI of the remote peer. This route can be used to test connectivity.

Usage:

```
Orthanc$get_peers_id_system(id)
```

Arguments:

`id` (character) Identifier of the peer of interest.

Returns: System information about the peer.

Method `get_plugins()`**:** List plugins

List all the installed plugins

Usage:

```
Orthanc$get_plugins()
```

Returns: List containing the identifiers of the installed plugins.

Method `get_plugins_explorer.js()`**:** JavaScript extensions to Orthanc Explorer

Get the JavaScript extensions that are installed by all the plugins using the `OrthancPluginExtendOrthancExplorer()` function of the plugin SDK. This route is for internal use of Orthanc Explorer.

Usage:

```
Orthanc$get_plugins_explorer.js()
```

Returns: The JavaScript extensions.

Method `get_plugins_id()`**:** Get plugin

Get system information about the plugin whose identifier is provided in the URL

Usage:

```
Orthanc$get_plugins_id(id)
```

Arguments:

`id` (character) Identifier of the job of interest.

Returns: JSON object containing information about the plugin.

Method `get_queries()`**:** List query/retrieve operations

List the identifiers of all the query/retrieve operations on DICOM modalities, as initiated by calls to `/modalities/{id}/query`. The length of this list is bounded by the `QueryRetrieveSize` configuration option of Orthanc. https://orthanc.uclouvain.be/book/users/rest.html#performing-query-retrieve-c-find-and-find-with-rest

Usage:

```
Orthanc$get_queries()
```

Returns: List containing the identifiers.

Method `delete_queries_id()`**:** Delete a query

Delete the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$delete_queries_id(id)
```

Arguments:

`id` (character) Identifier of the query of interest.

Returns: Nothing, invisibly.

Method `get_queries_id()`**:** List operations on a query

List the available operations for the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id(id)
```

Arguments:

`id` (character) Identifier of the query of interest.

Returns: List containing the list of operations.

Method `get_queries_id_answers()`**:** List answers to a query

List the indices of all the available answers resulting from a query/retrieve operation on some DICOM modality, whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id_answers(id, params = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual answers
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: List containing the indices of the answers, or detailed information about the reported answers (if `expand` argument is provided).

Method `get_queries_id_answers_index()`**:** List operations on an answer

List the available operations on an answer associated with the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id_answers_index(id, index)
```

Arguments:

`id` (character) Identifier of the query of interest.

`index` (character) Index of the answer.

Returns: List containing the list of operations.

Method `get_queries_id_answers_index_content()`**:** Get one answer

Get the content (DICOM tags) of one answer associated with the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id_answers_index_content(id, index, params = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`index` (character) Index of the answer.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object containing the DICOM tags of the answer.

Method `post_queries_id_answers_index_query_instances()`**:** Query the child instances of an answer

Issue a second DICOM C-FIND operation, in order to query the child instances associated with one answer to some query/retrieve operation whose identifiers are provided in the URL

Usage:

```
Orthanc$post_queries_id_answers_index_query_instances(id, index, json = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`index` (character) Index of the answer.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Query (list): Associative array containing the filter on the values of the DICOM tags
- Timeout (numeric): Timeout for the C-FIND command, in seconds (new in Orthanc 1.9.1)

Returns: Nothing, invisibly.

Method `post_queries_id_answers_index_query_series()`**:** Query the child series of an answer

Issue a second DICOM C-FIND operation, in order to query the child series associated with one answer to some query/retrieve operation whose identifiers are provided in the URL

Usage:

```
Orthanc$post_queries_id_answers_index_query_series(id, index, json = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`index` (character) Index of the answer.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Query (list): Associative array containing the filter on the values of the DICOM tags

- Timeout (numeric): Timeout for the C-FIND command, in seconds (new in Orthanc 1.9.1)

Returns: Nothing, invisibly.

Method `post_queries_id_answers_index_query_studies()`**:** Query the child studies of an answer

Issue a second DICOM C-FIND operation, in order to query the child studies associated with one answer to some query/retrieve operation whose identifiers are provided in the URL

Usage:

```
Orthanc$post_queries_id_answers_index_query_studies(id, index, json = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`index` (character) Index of the answer.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Query (list): Associative array containing the filter on the values of the DICOM tags
- Timeout (numeric): Timeout for the C-FIND command, in seconds (new in Orthanc 1.9.1)

Returns: Nothing, invisibly.

Method `post_queries_id_answers_index_retrieve()`**:** Retrieve one answer with a C-MOVE or a C-GET SCU

Start a C-MOVE or a C-GET SCU command as a job, in order to retrieve one answer associated with the query/retrieve operation whose identifiers are provided in the URL: https://orthanc.uclouvain.be/book/users/rest.htm retrieve-c-move

Usage:

```
Orthanc$post_queries_id_answers_index_retrieve(
  id,
  index,
  json = NULL,
  data = NULL
)
```

Arguments:

`id` (character) Identifier of the query of interest.

`index` (character) Index of the answer.

`json` (list) Named-list for request body. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Full (logical): If set to `TRUE`, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.

- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- RetrieveMethod (character): Force usage of C-MOVE or C-GET to retrieve the resource. If note defined in the payload, the retrieve method is defined in the DicomDefaultRetrieveMethod configuration or in DicomModalities->..->RetrieveMethod
- Simplify (logical): If set to `TRUE`, report the DICOM tags in human-readable format (using the symbolic name of the tags)
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- TargetAet (character): AET of the target modality. By default, the AET of Orthanc is used, as defined in the `DicomAet` configuration option.
- Timeout (numeric): Timeout for the C-MOVE command, in seconds
- UserData (list): User data that will travel along with the job. Request body: AET of the target modality (text/plain).

Returns: Nothing, invisibly.

Method `get_queries_id_level()`**:** Get level of original query

Get the query level (value of the `QueryRetrieveLevel` tag) of the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id_level(id)
```

Arguments:

`id` (character) Identifier of the query of interest.

Returns: The level.

Method `get_queries_id_modality()`**:** Get modality of original query

Get the identifier of the DICOM modality that was targeted by the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id_modality(id)
```

Arguments:

`id` (character) Identifier of the query of interest.

Returns: The identifier of the DICOM modality.

Method `get_queries_id_query()`**:** Get original query arguments

Get the original DICOM filter associated with the query/retrieve operation whose identifier is provided in the URL

Usage:

```
Orthanc$get_queries_id_query(id, params = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: Content of the original query.

Method `post_queries_id_retrieve()`**:** Retrieve all answers with C-MOVE SCU

Start a C-MOVE SCU command as a job, in order to retrieve all the answers associated with the query/retrieve operation whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/rest.html#perform retrieve-c-move

Usage:

```
Orthanc$post_queries_id_retrieve(id, json = NULL, data = NULL)
```

Arguments:

`id` (character) Identifier of the query of interest.

`json` (list) Named-list for request body. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Full (logical): If set to `TRUE`, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- RetrieveMethod (character): Force usage of C-MOVE or C-GET to retrieve the resource. If note defined in the payload, the retrieve method is defined in the DicomDefaultRetrieveMethod configuration or in DicomModalities->..->RetrieveMethod
- Simplify (logical): If set to `TRUE`, report the DICOM tags in human-readable format (using the symbolic name of the tags)
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- TargetAet (character): AET of the target modality. By default, the AET of Orthanc is used, as defined in the `DicomAet` configuration option.
- Timeout (numeric): Timeout for the C-MOVE command, in seconds
- UserData (list): User data that will travel along with the job. Request body: AET of the target modality (text/plain).

Returns: Nothing, invisibly.

Method `get_series()`**:** List the available series

List the Orthanc identifiers of all the available DICOM series

Usage:

```
Orthanc$get_series(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual resources, not only their Orthanc identifiers
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- limit (number): Limit the number of results
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- response-content (string): Defines the content of response for each returned resource. Allowed values are `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`, `Attachments`. If not specified, Orthanc will return `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`.e.g: 'response-content=MainDicomTags;Children (new in Orthanc 1.12.5 - overrides `expand`)
- short (boolean): If present, report the DICOM tags in hexadecimal format
- since (number): Show only the resources since the provided index

Returns: List containing either the Orthanc identifiers, or detailed information about the reported series (if `expand` argument is provided).

Method `delete_series_id()`**:** Delete some series

Delete the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$delete_series_id(id)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

Returns: Nothing, invisibly.

Method `get_series_id()`**:** Get information about some series

Get detailed information about the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the DICOM series.

Method `post_series_id_anonymize()`**:** Anonymize series

Start a job that will anonymize all the DICOM instances within the series whose identifier is provided in the URL. The modified DICOM instances will be stored into a brand new series, whose Orthanc identifiers will be returned by the job. https://orthanc.uclouvain.be/book/users/anonymization.html#anonymization-of-patients-studies-or-series

Usage:

```
Orthanc$post_series_id_anonymize(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- DicomVersion (character): Version of the DICOM standard to be used for anonymization. Check out configuration option `DeidentifyLogsDicomVersion` for possible values.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): List of DICOM tags whose value must not be destroyed by the anonymization. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- KeepLabels (logical): Keep the labels of all resources level (defaults to `FALSE`)
- KeepPrivateTags (logical): Keep the private tags from the DICOM instances (defaults to `FALSE`)
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`

- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of additional tags to be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_series_id_archive()`**:** Create ZIP archive

Synchronously create a ZIP archive containing the DICOM series whose Orthanc identifier is provided in the URL. This flavor is synchronous, which might *not* be desirable to archive large amount of data, as it might lead to network timeouts. Prefer the asynchronous version using `POST` method.

Usage:

```
Orthanc$get_series_id_archive(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: ZIP file containing the archive.

Method `post_series_id_archive()`**:** Create ZIP archive

Create a ZIP archive containing the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$post_series_id_archive(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If TRUE, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Synchronous (logical): If TRUE, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_series_id_attachments()`**:** List attachments

Get the list of attachments that are associated with the given series

Usage:

```
Orthanc$get_series_id_attachments(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (string): If present, retrieve the attachments list and their numerical ids

Returns: List containing the names of the attachments.

Method `delete_series_id_attachments_name()`**:** Delete attachment

Delete an attachment associated with the given DICOM series. This call will fail if trying to delete a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_series_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the attachment, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_series_id_attachments_name()`**:** List operations on attachments

Get the list of the operations that are available for attachments associated with the given series

Usage:

```
Orthanc$get_series_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: List of the available operations.

Method `put_series_id_attachments_name()`**:** Set attachment

Attach a file to the given DICOM series. This call will fail if trying to modify a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_series_id_attachments_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Binary data containing the attachment (application/octet-stream).
Optional headers (`headers`):

- If-Match (string): Revision of the attachment, if this is not the first time this attachment is set.

Returns: Empty JSON object in the case of a success.

Method `post_series_id_attachments_name_compress()`**:** Compress attachment

Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_series_id_attachments_name_compress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `get_series_id_attachments_name_compressed_data()`: Get attachment (no decompression)

Get the (binary) content of one attachment associated with the given series. The attachment will not be decompressed if `StorageCompression` is `TRUE`.

Usage:

```
Orthanc$get_series_id_attachments_name_compressed_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_series_id_attachments_name_compressed_md5()`: Get MD5 of attachment on disk

Get the MD5 hash of one attachment associated with the given series, as stored on the disk. This is different from `.../md5` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_series_id_attachments_name_compressed_md5(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment, as stored on the disk.

Method `get_series_id_attachments_name_compressed_size()`**:** Get size of attachment on disk

Get the size of one attachment associated with the given series, as stored on the disk. This is different from `.../size` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_series_id_attachments_name_compressed_size(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment, as stored on the disk.

Method `get_series_id_attachments_name_data()`**:** Get attachment

Get the (binary) content of one attachment associated with the given series

Usage:

```
Orthanc$get_series_id_attachments_name_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_series_id_attachments_name_info()`**:** Get info about the attachment
Get all the information about the attachment associated with the given series

Usage:

```
Orthanc$get_series_id_attachments_name_info(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: JSON object containing the information about the attachment.

Method `get_series_id_attachments_name_is_compressed()`**:** Is attachment compressed?
Test whether the attachment has been stored as a compressed file on the disk.

Usage:

```
Orthanc$get_series_id_attachments_name_is_compressed(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: `0` if the attachment was stored uncompressed, `1` if it was compressed.

Method `get_series_id_attachments_name_md5()`**:** Get MD5 of attachment
Get the MD5 hash of one attachment associated with the given series

Usage:

```
Orthanc$get_series_id_attachments_name_md5(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment.

Method `get_series_id_attachments_name_size()`**:** Get size of attachment

Get the size of one attachment associated with the given series

Usage:

```
Orthanc$get_series_id_attachments_name_size(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment.

Method `post_series_id_attachments_name_uncompress()`**:** Uncompress attachment

Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_series_id_attachments_name_uncompress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `post_series_id_attachments_name_verify_md5()`**:** Verify attachment

Verify that the attachment is not corrupted, by validating its MD5 hash

Usage:

```
Orthanc$post_series_id_attachments_name_verify_md5(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: On success, a valid JSON object is returned.

Method `get_series_id_instances()`**:** Get child instances

Get detailed information about the child instances of the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id_instances(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If false or missing, only retrieve the list of child instances
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: List containing information about the child DICOM instances.

Method `get_series_id_instances_tags()`**:** Get tags of instances

Get the tags of all the child instances of the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id_instances_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object associating the Orthanc identifiers of the instances, with the values of their DICOM tags.

Method `get_series_id_labels()`**:** List labels

Get the labels that are associated with the given series (new in Orthanc 1.12.0)

Usage:

```
Orthanc$get_series_id_labels(id)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

Returns: List containing the names of the labels.

Method `delete_series_id_labels_label()`**:** Remove label

Remove a label associated with a series

Usage:

```
Orthanc$delete_series_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`label` (character) The label to be removed.

Returns: Nothing, invisibly.

Method `get_series_id_labels_label()`**:** Test label

Test whether the series is associated with the given label

Usage:

```
Orthanc$get_series_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`label` (character) The label of interest.

Returns: Empty string is returned in the case of presence, error 404 in the case of absence.

Method `put_series_id_labels_label()`**:** Add label

Associate a label with a series

Usage:

```
Orthanc$put_series_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`label` (character) The label to be added.

Returns: Nothing, invisibly.

Method `get_series_id_media()`**:** Create DICOMDIR media

Synchronously create a DICOMDIR media containing the DICOM series whose Orthanc identifier is provided in the URL. This flavor is synchronous, which might *not* be desirable to archive large amount of data, as it might lead to network timeouts. Prefer the asynchronous version using `POST` method.

Usage:

```
Orthanc$get_series_id_media(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- extended (string): If present, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*
- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)

- transcode (string): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: ZIP file containing the archive.

Method `post_series_id_media()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$post_series_id_media(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Extended (logical): If `TRUE`, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*. Default value is `FALSE`.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_series_id_metadata()`**:** List metadata

Get the list of metadata that are associated with the given series

Usage:

```
Orthanc$get_series_id_metadata(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, also retrieve the value of the individual metadata
- numeric (string): If present, use the numeric identifier of the metadata instead of its symbolic name

Returns: List containing the names of the available metadata, or List mapping metadata to their values (if `expand` argument is provided).

Method `delete_series_id_metadata_name()`**:** Delete metadata

Delete some metadata associated with the given DICOM series. This call will fail if trying to delete a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_series_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the metadata, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_series_id_metadata_name()`**:** Get metadata

Get the value of a metadata that is associated with the given series

Usage:

```
Orthanc$get_series_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the metadata, to check if its content has changed

Returns: Value of the metadata.

Method `put_series_id_metadata_name()`**:** Set metadata

Set the value of some metadata in the given DICOM series. This call will fail if trying to modify a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_series_id_metadata_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: String value of the metadata (text/plain).
Optional headers (`headers`):

- If-Match (string): Revision of the metadata, if this is not the first time this metadata is set.

Returns: Nothing, invisibly.

Method `post_series_id_modify()`**:** Modify series

Start a job that will modify all the DICOM instances within the series whose identifier is provided in the URL. The modified DICOM instances will be stored into a brand new series, whose Orthanc identifiers will be returned by the job. https://orthanc.uclouvain.be/book/users/anonymization.html#modification-of-studies-or-series

Usage:

```
Orthanc$post_series_id_modify(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): Keep the original value of the specified tags, to be chosen among the `StudyInstanceUID`, `SeriesInstanceUID` and `SOPInstanceUID` tags. Avoid this feature as much as possible, as this breaks the DICOM model of the real world.
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of tags that must be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- RemovePrivateTags (logical): Remove the private tags from the DICOM instances (defaults to `FALSE`)

- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_series_id_module()`**:** Get series module

Get the series module of the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id_module(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: Information about the DICOM series.

Method `get_series_id_numpy()`**:** Decode series for numpy

Decode the given DICOM series, for use with numpy in Python. The numpy array has 4 dimensions: (frame, height, width, color channel).

Usage:

```
Orthanc$get_series_id_numpy(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the DICOM resource of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- compress (boolean): Compress the file as `.npz`
- rescale (boolean): On grayscale images, apply the rescaling and return floating-point values

Returns: Numpy file: https://numpy.org/devdocs/reference/generated/numpy.lib.format.html.

Method `get_series_id_patient()`**:** Get parent patient

Get detailed information about the parent patient of the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id_patient(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the parent DICOM patient.

Method `post_series_id_reconstruct()`**:** Reconstruct tags & optionally files of series

Reconstruct the main DICOM tags in DB of the series whose Orthanc identifier is provided in the URL. This is useful if child studies/series/instances have inconsistent values for higher-level tags, in order to force Orthanc to use the value from the resource of interest. Beware that this is a time-consuming operation, as all the children DICOM instances will be parsed again, and the Orthanc index will be updated accordingly.

Usage:

```
Orthanc$post_series_id_reconstruct(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- LimitToThisLevelMainDicomTags (logical): Only reconstruct this level MainDicomTags by re-reading them from a random child instance of the resource. This option is much faster than a full reconstruct and is useful e.g. if you have modified the 'ExtraMainDicomTags' at the Study level to optimize the speed of some C-Find. 'false' by default. (New in Orthanc 1.12.4)
- ReconstructFiles (logical): Also reconstruct the files of the resources (e.g: apply Ingest-Transcoding, StorageCompression). 'false' by default. (New in Orthanc 1.11.0)

Returns: Nothing, invisibly.

Method `get_series_id_shared_tags()`**:** Get shared tags

Extract the DICOM tags whose value is constant across all the child instances of the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id_shared_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object containing the values of the DICOM tags.

Method `get_series_id_statistics()`**:** Get series statistics

Get statistics about the given series

Usage:

```
Orthanc$get_series_id_statistics(id)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

Returns: Nothing, invisibly.

Method `get_series_id_study()`**:** Get parent study

Get detailed information about the parent study of the DICOM series whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_series_id_study(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the series of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the parent DICOM study.

Method `get_statistics()`**:** Get database statistics

Get statistics related to the database of Orthanc

Usage:

```
Orthanc$get_statistics()
```

Returns: Nothing, invisibly.

Method `get_storage_commitment_id()`**:** Get storage commitment report

Get the storage commitment report whose identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/storage commitment.html#storage-commitment-scu

Usage:

```
Orthanc$get_storage_commitment_id(id)
```

Arguments:

`id` (character) Identifier of the storage commitment report.

Returns: Nothing, invisibly.

Method `post_storage_commitment_id_remove()`**:** Remove after storage commitment

Remove out of Orthanc, the DICOM instances that have been reported to have been properly received in the storage commitment report whose identifier is provided in the URL. This is only possible if the `Status` of the storage commitment report is `Success`. https://orthanc.uclouvain.be/book/users/storage-commitment.html#removing-the-instances

Usage:

```
Orthanc$post_storage_commitment_id_remove(id)
```

Arguments:

`id` (character) Identifier of the storage commitment report.

Returns: Nothing, invisibly.

Method `get_studies()`**:** List the available studies

List the Orthanc identifiers of all the available DICOM studies

Usage:

```
Orthanc$get_studies(params = NULL)
```

Arguments:

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, retrieve detailed information about the individual resources, not only their Orthanc identifiers
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- limit (number): Limit the number of results
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.

- response-content (string): Defines the content of response for each returned resource. Allowed values are `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`, `Attachments`. If not specified, Orthanc will return `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`.e.g: 'response-content=MainDicomTags;Children (new in Orthanc 1.12.5 - overrides `expand`)
- short (boolean): If present, report the DICOM tags in hexadecimal format
- since (number): Show only the resources since the provided index

Returns: List containing either the Orthanc identifiers, or detailed information about the reported studies (if `expand` argument is provided).

Method `delete_studies_id()`**:** Delete some study

Delete the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$delete_studies_id(id)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

Returns: Nothing, invisibly.

Method `get_studies_id()`**:** Get information about some study

Get detailed information about the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the DICOM study.

Method `post_studies_id_anonymize()`**:** Anonymize study

Start a job that will anonymize all the DICOM instances within the study whose identifier is provided in the URL. The modified DICOM instances will be stored into a brand new study, whose Orthanc identifiers will be returned by the job. https://orthanc.uclouvain.be/book/users/anonymization.html#anonymization-of-patients-studies-or-series

Usage:

```
Orthanc$post_studies_id_anonymize(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- DicomVersion (character): Version of the DICOM standard to be used for anonymization. Check out configuration option `DeidentifyLogsDicomVersion` for possible values.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): List of DICOM tags whose value must not be destroyed by the anonymization. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- KeepLabels (logical): Keep the labels of all resources level (defaults to `FALSE`)
- KeepPrivateTags (logical): Keep the private tags from the DICOM instances (defaults to `FALSE`)
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of additional tags to be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_studies_id_archive()`**:** Create ZIP archive

Synchronously create a ZIP archive containing the DICOM study whose Orthanc identifier is provided in the URL. This flavor is synchronous, which might *not* be desirable to archive large

amount of data, as it might lead to network timeouts. Prefer the asynchronous version using `POST` method.

Usage:

```
Orthanc$get_studies_id_archive(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: ZIP file containing the archive.

Method `post_studies_id_archive()`**:** Create ZIP archive

Create a ZIP archive containing the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$post_studies_id_archive(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_studies_id_attachments()`**:** List attachments

Get the list of attachments that are associated with the given study

Usage:

```
Orthanc$get_studies_id_attachments(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (string): If present, retrieve the attachments list and their numerical ids

Returns: List containing the names of the attachments.

Method `delete_studies_id_attachments_name()`**:** Delete attachment

Delete an attachment associated with the given DICOM study. This call will fail if trying to delete a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_studies_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the attachment, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_studies_id_attachments_name()`**:** List operations on attachments

Get the list of the operations that are available for attachments associated with the given study

Usage:

```
Orthanc$get_studies_id_attachments_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: List of the available operations.

Method `put_studies_id_attachments_name()`: Set attachment

Attach a file to the given DICOM study. This call will fail if trying to modify a system attachment (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_studies_id_attachments_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Binary data containing the attachment (application/octet-stream). Optional headers (`headers`):

- If-Match (string): Revision of the attachment, if this is not the first time this attachment is set.

Returns: Empty JSON object in the case of a success.

Method `post_studies_id_attachments_name_compress()`: Compress attachment

Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_studies_id_attachments_name_compress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `get_studies_id_attachments_name_compressed_data()`: Get attachment (no decompression)

Get the (binary) content of one attachment associated with the given study. The attachment will not be decompressed if `StorageCompression` is `TRUE`.

Usage:

```
Orthanc$get_studies_id_attachments_name_compressed_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_studies_id_attachments_name_compressed_md5()`**:** Get MD5 of attachment on disk

Get the MD5 hash of one attachment associated with the given study, as stored on the disk. This is different from `.../md5` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_studies_id_attachments_name_compressed_md5(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment, as stored on the disk.

Method `get_studies_id_attachments_name_compressed_size()`**:** Get size of attachment on disk

Get the size of one attachment associated with the given study, as stored on the disk. This is different from `.../size` iff `EnableStorage` is `TRUE`.

Usage:

```
Orthanc$get_studies_id_attachments_name_compressed_size(
  id,
  name,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment, as stored on the disk.

Method `get_studies_id_attachments_name_data()`**:** Get attachment

Get the (binary) content of one attachment associated with the given study

Usage:

```
Orthanc$get_studies_id_attachments_name_data(
  id,
  name,
  params = NULL,
  headers = NULL
)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`params` (list) Named-list of optional query parameters. See Details.

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- Content-Range (string): Optional content range to access part of the attachment (new in Orthanc 1.12.5)
- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)

Returns: The attachment.

Method `get_studies_id_attachments_name_info()`**:** Get info about the attachment

Get all the information about the attachment associated with the given study

Usage:

```
Orthanc$get_studies_id_attachments_name_info(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: JSON object containing the information about the attachment.

Method `get_studies_id_attachments_name_is_compressed()`**:** Is attachment compressed?
Test whether the attachment has been stored as a compressed file on the disk.

Usage:

```
Orthanc$get_studies_id_attachments_name_is_compressed(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: `0` if the attachment was stored uncompressed, `1` if it was compressed.

Method `get_studies_id_attachments_name_md5()`**:** Get MD5 of attachment
Get the MD5 hash of one attachment associated with the given study

Usage:

```
Orthanc$get_studies_id_attachments_name_md5(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The MD5 of the attachment.

Method `get_studies_id_attachments_name_size()`**:** Get size of attachment
Get the size of one attachment associated with the given study

Usage:

```
Orthanc$get_studies_id_attachments_name_size(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the attachment, to check if its content has changed

Returns: The size of the attachment.

Method `post_studies_id_attachments_name_uncompress()`**:** Uncompress attachment
Change the compression scheme that is used to store an attachment.

Usage:

```
Orthanc$post_studies_id_attachments_name_uncompress(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: Nothing, invisibly.

Method `post_studies_id_attachments_name_verify_md5()`**:** Verify attachment
Verify that the attachment is not corrupted, by validating its MD5 hash

Usage:

```
Orthanc$post_studies_id_attachments_name_verify_md5(id, name)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the attachment, or its index (cf. `UserContentType` configuration option).

Returns: On success, a valid JSON object is returned.

Method `get_studies_id_instances()`**:** Get child instances
Get detailed information about the child instances of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_instances(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If false or missing, only retrieve the list of child instances
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.

- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: List containing information about the child DICOM instances.

Method `get_studies_id_instances_tags()`**:** Get tags of instances

Get the tags of all the child instances of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_instances_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object associating the Orthanc identifiers of the instances, with the values of their DICOM tags.

Method `get_studies_id_labels()`**:** List labels

Get the labels that are associated with the given study (new in Orthanc 1.12.0)

Usage:

```
Orthanc$get_studies_id_labels(id)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

Returns: List containing the names of the labels.

Method `delete_studies_id_labels_label()`**:** Remove label

Remove a label associated with a study

Usage:

```
Orthanc$delete_studies_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`label` (character) The label to be removed.

Returns: Nothing, invisibly.

Method `get_studies_id_labels_label()`**:** Test label

Test whether the study is associated with the given label

Usage:

```
Orthanc$get_studies_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`label` (character) The label of interest.

Returns: Empty string is returned in the case of presence, error 404 in the case of absence.

Method `put_studies_id_labels_label()`**:** Add label

Associate a label with a study

Usage:

```
Orthanc$put_studies_id_labels_label(id, label)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`label` (character) The label to be added.

Returns: Nothing, invisibly.

Method `get_studies_id_media()`**:** Create DICOMDIR media

Synchronously create a DICOMDIR media containing the DICOM study whose Orthanc identifier is provided in the URL. This flavor is synchronous, which might *not* be desirable to archive large amount of data, as it might lead to network timeouts. Prefer the asynchronous version using `POST` method.

Usage:

```
Orthanc$get_studies_id_media(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- extended (string): If present, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*
- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: ZIP file containing the archive.

Method `post_studies_id_media()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$post_studies_id_media(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If TRUE, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Extended (logical): If TRUE, will include additional tags such as SeriesDescription, leading to a so-called *extended DICOMDIR*. Default value is FALSE.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Synchronous (logical): If TRUE, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `post_studies_id_merge()`**:** Merge study

Start a new job so as to move some DICOM resources into the DICOM study whose Orthanc identifier is provided in the URL: https://orthanc.uclouvain.be/book/users/anonymization.html#merging

Usage:

```
Orthanc$post_studies_id_merge(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If TRUE, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- KeepSource (logical): If set to TRUE, instructs Orthanc to keep a copy of the original resources in their source study. By default, the original resources are deleted from Orthanc.
- Permissive (logical): If TRUE, ignore errors during the individual steps of the job. Default value is FALSE.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is 0
- Resources (list): The list of DICOM resources (studies, series, and/or instances) to be merged into the study of interest (mandatory option)
- Synchronous (logical): If TRUE, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.

- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_studies_id_metadata()`**:** List metadata

Get the list of metadata that are associated with the given study

Usage:

```
Orthanc$get_studies_id_metadata(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If present, also retrieve the value of the individual metadata
- numeric (string): If present, use the numeric identifier of the metadata instead of its symbolic name

Returns: List containing the names of the available metadata, or List mapping metadata to their values (if `expand` argument is provided).

Method `delete_studies_id_metadata_name()`**:** Delete metadata

Delete some metadata associated with the given DICOM study. This call will fail if trying to delete a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$delete_studies_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-Match (string): Revision of the metadata, to check if its content has not changed and can be deleted. This header is mandatory if `CheckRevisions` option is `TRUE`.

Returns: Nothing, invisibly.

Method `get_studies_id_metadata_name()`**:** Get metadata

Get the value of a metadata that is associated with the given study

Usage:

```
Orthanc$get_studies_id_metadata_name(id, name, headers = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

Details: Optional headers (`headers`):

- If-None-Match (string): Optional revision of the metadata, to check if its content has changed

Returns: Value of the metadata.

Method `put_studies_id_metadata_name()`**:** Set metadata

Set the value of some metadata in the given DICOM study. This call will fail if trying to modify a system metadata (i.e. whose index is < 1024).

Usage:

```
Orthanc$put_studies_id_metadata_name(id, name, headers = NULL, data = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`name` (character) The name of the metadata, or its index (cf. `UserMetadata` configuration option).

`headers` (list) Named-list of optional header parameters. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: String value of the metadata (text/plain).
Optional headers (`headers`):

- If-Match (string): Revision of the metadata, if this is not the first time this metadata is set.

Returns: Nothing, invisibly.

Method `post_studies_id_modify()`**:** Modify study

Start a job that will modify all the DICOM instances within the study whose identifier is provided in the URL. The modified DICOM instances will be stored into a brand new study, whose Orthanc identifiers will be returned by the job. https://orthanc.uclouvain.be/book/users/anonymization.html#modification-of-studies-or-series

Usage:

```
Orthanc$post_studies_id_modify(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): Keep the original value of the specified tags, to be chosen among the `StudyInstanceUID`, `SeriesInstanceUID` and `SOPInstanceUID` tags. Avoid this feature as much as possible, as this breaks the DICOM model of the real world.
- KeepSource (logical): If set to `FALSE`, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)

- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- PrivateCreator (character): The private creator to be used for private tags in `Replace`
- Remove (list): List of tags that must be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- RemovePrivateTags (logical): Remove the private tags from the DICOM instances (defaults to `FALSE`)
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the `dcmodify` command-line tool (wildcards are supported as well).
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_studies_id_module()`**:** Get study module

Get the study module of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_module(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: Information about the DICOM study.

Method `get_studies_id_module_patient()`**:** Get patient module of study

Get the patient module of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_module_patient(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- ignore-length (array): Also include the DICOM tags that are provided in this list, even if their associated value is long
- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: Information about the DICOM study.

Method `get_studies_id_patient()`**:** Get parent patient

Get detailed information about the parent patient of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_patient(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'Requested-Tags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: Information about the parent DICOM patient.

Method `post_studies_id_reconstruct()`**:** Reconstruct tags & optionally files of study

Reconstruct the main DICOM tags in DB of the study whose Orthanc identifier is provided in the URL. This is useful if child studies/series/instances have inconsistent values for higher-level tags, in order to force Orthanc to use the value from the resource of interest. Beware that this is a time-consuming operation, as all the children DICOM instances will be parsed again, and the Orthanc index will be updated accordingly.

Usage:

```
Orthanc$post_studies_id_reconstruct(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- LimitToThisLevelMainDicomTags (logical): Only reconstruct this level MainDicomTags by re-reading them from a random child instance of the resource. This option is much faster than a full reconstruct and is useful e.g. if you have modified the 'ExtraMainDicomTags' at the Study level to optimize the speed of some C-Find. 'false' by default. (New in Orthanc 1.12.4)

- ReconstructFiles (logical): Also reconstruct the files of the resources (e.g: apply IngestTranscoding, StorageCompression). 'false' by default. (New in Orthanc 1.11.0)

Returns: Nothing, invisibly.

Method `get_studies_id_series()`**:** Get child series

Get detailed information about the child series of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_series(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- expand (string): If false or missing, only retrieve the list of child series
- full (boolean): If present, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- requested-tags (string): If present, list the DICOM Tags you want to list in the response. This argument is a semi-column separated list of DICOM Tags identifiers; e.g: 'requested-tags=0010,0010;PatientBirthDate'. The tags requested tags are returned in the 'RequestedTags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- short (boolean): If present, report the DICOM tags in hexadecimal format

Returns: List containing information about the child DICOM series.

Method `get_studies_id_shared_tags()`**:** Get shared tags

Extract the DICOM tags whose value is constant across all the child instances of the DICOM study whose Orthanc identifier is provided in the URL

Usage:

```
Orthanc$get_studies_id_shared_tags(id, params = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- short (boolean): If present, report the DICOM tags in hexadecimal format
- simplify (boolean): If present, report the DICOM tags in human-readable format (using the symbolic name of the tags)

Returns: JSON object containing the values of the DICOM tags.

Method `post_studies_id_split()`**:** Split study

Start a new job so as to split the DICOM study whose Orthanc identifier is provided in the URL, by taking some of its children series or instances out of it and putting them into a brand new study (this new study is created by setting the `StudyInstanceUID` tag to a random identifier): https://orthanc.uclouvain.be/book/users/anonymization.html#splitting

Usage:

```
Orthanc$post_studies_id_split(id, json = NULL)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Instances (list): The list of instances to be separated from the parent study. These instances must all be children of the same source study, that is specified in the URI.
- KeepLabels (logical): Keep the labels of all resources level (defaults to `FALSE`)
- KeepSource (logical): If set to `TRUE`, instructs Orthanc to keep a copy of the original series/instances in the source study. By default, the original series/instances are deleted from Orthanc.
- Permissive (logical): If `TRUE`, ignore errors during the individual steps of the job. Default value is `FALSE`.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is `0`
- Remove (list): List of tags that must be removed in the new study (from the same modules as in the `Replace` option)
- Replace (list): Associative array to change the value of some DICOM tags in the new study. These tags must be part of the "Patient Module Attributes" or the "General Study Module Attributes", as specified by the DICOM 2011 standard in Tables C.7-1 and C.7-3.
- Series (list): The list of series to be separated from the parent study. These series must all be children of the same source study, that is specified in the URI.
- Synchronous (logical): If `TRUE`, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- UserData (list): User data that will travel along with the job.

Returns: Nothing, invisibly.

Method `get_studies_id_statistics()`**:** Get study statistics

Get statistics about the given study

Usage:

```
Orthanc$get_studies_id_statistics(id)
```

Arguments:

`id` (character) Orthanc identifier of the study of interest.

Returns: Nothing, invisibly.

Method `get_system()`**:** Get system information

Get system information about Orthanc

Usage:

```
Orthanc$get_system()
```

Returns: Nothing, invisibly.

Method `get_tools()`**:** List operations
List the available operations under URI `/tools/`

Usage:

```
Orthanc$get_tools()
```

Returns: List of the available operations.

Method `get_tools_accepted_sop_classes()`**:** Get accepted SOPClassUID
Get the list of SOP Class UIDs that are accepted by Orthanc C-STORE SCP. This corresponds to the configuration options `AcceptedSopClasses` and `RejectedSopClasses`.

Usage:

```
Orthanc$get_tools_accepted_sop_classes()
```

Returns: List containing the SOP Class UIDs.

Method `get_tools_accepted_transfer_syntaxes()`**:** Get accepted transfer syntaxes
Get the list of UIDs of the DICOM transfer syntaxes that are accepted by Orthanc C-STORE SCP.
This corresponds to the configuration options `AcceptedTransferSyntaxes` and `XXXTransferSyntaxAccepted`.

Usage:

```
Orthanc$get_tools_accepted_transfer_syntaxes()
```

Returns: List containing the transfer syntax UIDs.

Method `put_tools_accepted_transfer_syntaxes()`**:** Set accepted transfer syntaxes
Set the DICOM transfer syntaxes that accepted by Orthanc C-STORE SCP

Usage:

```
Orthanc$put_tools_accepted_transfer_syntaxes(json = NULL, data = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body JSON schema (application/json):

- (list): JSON array containing a list of transfer syntax UIDs to be accepted. Wildcards ? and * are accepted. Request body: UID of the transfer syntax to be accepted. Wildcards ? and * are accepted. (text/plain).

Returns: List containing the now-accepted transfer syntax UIDs.

Method `post_tools_bulk_anonymize()`**:** Anonymize a set of resources
Start a job that will anonymize all the DICOM patients, studies, series or instances whose identifiers are provided in the `Resources` field.

Usage:

```
Orthanc$post_tools_bulk_anonymize(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If TRUE, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- DicomVersion (character): Version of the DICOM standard to be used for anonymization. Check out configuration option DeidentifyLogsDicomVersion for possible values.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): List of DICOM tags whose value must not be destroyed by the anonymization. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the dcmodify command-line tool (wildcards are supported as well).
- KeepLabels (logical): Keep the labels of all resources level (defaults to FALSE)
- KeepPrivateTags (logical): Keep the private tags from the DICOM instances (defaults to FALSE)
- KeepSource (logical): If set to FALSE, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If TRUE, ignore errors during the individual steps of the job. Default value is FALSE.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is 0
- PrivateCreator (character): The private creator to be used for private tags in Replace
- Remove (list): List of additional tags to be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the dcmodify command-line tool (wildcards are supported as well).
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the dcmodify command-line tool (wildcards are supported as well).
- Resources (list): List of the Orthanc identifiers of the patients/studies/series/instances of interest.
- Synchronous (logical): If TRUE, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: The list of all the resources that have been created by this anonymization.

Method `post_tools_bulk_content()`**:** Describe a set of resources

Get the content all the DICOM patients, studies, series or instances whose identifiers are provided in the Resources field, in one single call.

Usage:

```
Orthanc$post_tools_bulk_content(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Full (logical): If set to `TRUE`, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- Level (character): This optional argument specifies the level of interest (can be `Patient`, `Study`, `Series` or `Instance`). Orthanc will loop over the items inside `Resources`, and explore upward or downward in the DICOM hierarchy in order to find the level of interest.
- Metadata (logical): If set to `TRUE` (default value), the metadata associated with the resources will also be retrieved.
- Resources (list): List of the Orthanc identifiers of the patients/studies/series/instances of interest.
- Short (logical): If set to `TRUE`, report the DICOM tags in hexadecimal format

Returns: Nothing, invisibly.

Method `post_tools_bulk_delete()`**:** Delete a set of resources

Delete all the DICOM patients, studies, series or instances whose identifiers are provided in the `Resources` field.

Usage:

```
Orthanc$post_tools_bulk_delete(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Resources (list): List of the Orthanc identifiers of the patients/studies/series/instances of interest.

Returns: Nothing, invisibly.

Method `post_tools_bulk_modify()`**:** Modify a set of resources

Start a job that will modify all the DICOM patients, studies, series or instances whose identifiers are provided in the `Resources` field.

Usage:

```
Orthanc$post_tools_bulk_modify(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, run the job in asynchronous mode, which means that the REST API call will immediately return, reporting the identifier of a job. Prefer this flavor wherever possible.
- Force (logical): Allow the modification of tags related to DICOM identifiers, at the risk of breaking the DICOM model of the real world
- Keep (list): Keep the original value of the specified tags, to be chosen among the `StudyInstanceUID`, `SeriesInstanceUID` and `SOPInstanceUID` tags. Avoid this feature as much as possible, as this breaks the DICOM model of the real world.

- KeepSource (logical): If set to FALSE, instructs Orthanc to the remove original resources. By default, the original resources are kept in Orthanc.
- Level (character): Level of the modification (Patient, Study, Series or Instance). If absent, the level defaults to Instance, but is set to Patient if PatientID is modified, to Study if StudyInstanceUID is modified, or to Series if SeriesInstancesUID is modified. (new in Orthanc 1.9.7)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Permissive (logical): If TRUE, ignore errors during the individual steps of the job. Default value is FALSE.
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority. Default value is 0
- PrivateCreator (character): The private creator to be used for private tags in Replace
- Remove (list): List of tags that must be removed from the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the dcmodify command-line tool (wildcards are supported as well).
- RemovePrivateTags (logical): Remove the private tags from the DICOM instances (defaults to FALSE)
- Replace (list): Associative array to change the value of some DICOM tags in the DICOM instances. Starting with Orthanc 1.9.4, paths to subsequences can be provided using the same syntax as the dcmodify command-line tool (wildcards are supported as well).
- Resources (list): List of the Orthanc identifiers of the patients/studies/series/instances of interest.
- Synchronous (logical): If TRUE, run the job in synchronous mode, which means that the HTTP answer will directly contain the result of the job. This is the default, easy behavior, but it is *not* desirable for long jobs, as it might lead to network timeouts.
- Transcode (character): Transcode the DICOM instances to the provided DICOM transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): User data that will travel along with the job.

Returns: The list of all the resources that have been altered by this modification.

Method `post_tools_count_resources()`**:** Count local resources

This URI can be used to count the resources that are matching criteria on the content of the local Orthanc server, in a way that is similar to tools/find

Usage:

```
Orthanc$post_tools_count_resources(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Full (logical): If set to TRUE, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- Labels (list): List of strings specifying which labels to look for in the resources (new in Orthanc 1.12.0)

- LabelsConstraint (character): Constraint on the labels, can be `All`, `Any`, or `None` (defaults to `All`, new in Orthanc 1.12.0)
- Level (character): Level of the query (`Patient`, `Study`, `Series` or `Instance`)
- MetadataQuery (list): Associative array containing the filter on the values of the metadata (new in Orthanc 1.12.5)
- ParentPatient (character): Limit the reported resources to descendants of this patient (new in Orthanc 1.12.5)
- ParentSeries (character): Limit the reported resources to descendants of this series (new in Orthanc 1.12.5)
- ParentStudy (character): Limit the reported resources to descendants of this study (new in Orthanc 1.12.5)
- Query (list): Associative array containing the filter on the values of the DICOM tags
- Short (logical): If set to `TRUE`, report the DICOM tags in hexadecimal format

Returns: A JSON object with the `Count` of matching resources.

Method `get_tools_create_archive()`**:** Create ZIP archive

Create a ZIP archive containing the DICOM resources (patients, studies, series, or instances) whose Orthanc identifiers are provided in the 'resources' argument

Usage:

```
Orthanc$get_tools_create_archive(resources = NULL, params = NULL)
```

Arguments:

`resources` (character) A comma separated list of Orthanc resource identifiers to include in the ZIP archive..

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: Nothing, invisibly.

Method `post_tools_create_archive()`**:** Create ZIP archive

Create a ZIP archive containing the DICOM resources (patients, studies, series, or instances) whose Orthanc identifiers are provided in the body

Usage:

```
Orthanc$post_tools_create_archive(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.

- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Resources (list): The list of Orthanc identifiers of interest.
- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `post_tools_create_dicom()`**:** Create one DICOM instance

Create one DICOM instance, and store it into Orthanc

Usage:

```
Orthanc$post_tools_create_dicom(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Content (character): This field can be used to embed an image (pixel data encoded as PNG or JPEG), a PDF, or a 3D manufactoring model (MTL/OBJ/STL) inside the created DICOM instance. The file to be encapsulated must be provided using its data URI scheme encoding. This field can possibly contain a JSON array, in which case a DICOM series is created containing one DICOM instance for each item in the `Content` field.
- Encapsulate (logical): If set to `TRUE`, encapsulate the binary data of `ContentData` as such, using a compressed transfer syntax. Only applicable if `ContentData` contains a grayscale or color JPEG image in 8bpp, in which case the transfer syntax is set to "1.2.840.10008.1.2.4.50". (new in Orthanc 1.12.7)
- Force (logical): Avoid the consistency checks for the DICOM tags that enforce the DICOM model of the real-world. You can notably use this flag if you need to manually set the tags `StudyInstanceUID`, `SeriesInstanceUID`, or `SOPInstanceUID`. Be careful with this feature.
- InterpretBinaryTags (logical): If some value in the `Tags` associative array is formatted according to some data URI scheme encoding, whether this value is decoded to a binary value or kept as such (`TRUE` by default)
- Parent (character): If present, the newly created instance will be attached to the parent DICOM resource whose Orthanc identifier is contained in this field. The DICOM tags of the parent modules in the DICOM hierarchy will be automatically copied to the newly created instance.

- PrivateCreator (character): The private creator to be used for private tags in `Tags`
- Tags (list): Associative array containing the tags of the new instance to be created

Returns: Nothing, invisibly.

Method `get_tools_create_media()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM resources (patients, studies, series, or instances) whose Orthanc identifiers are provided in the 'resources' argument

Usage:

```
Orthanc$get_tools_create_media(resources = NULL, params = NULL)
```

Arguments:

`resources` (character) A comma separated list of Orthanc resource identifiers to include in the DICOMDIR media..

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: Nothing, invisibly.

Method `post_tools_create_media()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM resources (patients, studies, series, or instances) whose Orthanc identifiers are provided in the body

Usage:

```
Orthanc$post_tools_create_media(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Extended (logical): If `TRUE`, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*. Default value is `FALSE`.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Resources (list): The list of Orthanc identifiers of interest.

- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_tools_create_media_extended()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM resources (patients, studies, series, or instances) whose Orthanc identifiers are provided in the 'resources' argument

Usage:

```
Orthanc$get_tools_create_media_extended(resources = NULL, params = NULL)
```

Arguments:

`resources` (character) A comma separated list of Orthanc resource identifiers to include in the DICOMDIR media..

`params` (list) Named-list of optional query parameters. See Details.

Details: Optional query parameters (`params`):

- filename (string): Filename to set in the "Content-Disposition" HTTP header (including file extension)
- lossy-quality (number): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- transcode (string): If present, the DICOM files will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html

Returns: Nothing, invisibly.

Method `post_tools_create_media_extended()`**:** Create DICOMDIR media

Create a DICOMDIR media containing the DICOM resources (patients, studies, series, or instances) whose Orthanc identifiers are provided in the body

Usage:

```
Orthanc$post_tools_create_media_extended(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- Asynchronous (logical): If `TRUE`, create the archive in asynchronous mode, which means that a job is submitted to create the archive in background.
- Extended (logical): If `TRUE`, will include additional tags such as `SeriesDescription`, leading to a so-called *extended DICOMDIR*. Default value is `TRUE`.
- Filename (character): Filename to set in the "Content-Disposition" HTTP header (including file extension)

- LossyQuality (numeric): If transcoding to a lossy transfer syntax, this entry defines the quality as an integer between 1 and 100. If not provided, the value is defined by the "DicomLossyTranscodingQuality" configuration. (new in v1.12.7)
- Priority (numeric): In asynchronous mode, the priority of the job. The higher the value, the higher the priority.
- Resources (list): The list of Orthanc identifiers of interest.
- Synchronous (logical): If `TRUE`, create the archive in synchronous mode, which means that the HTTP answer will directly contain the ZIP file. This is the default, easy behavior. However, if global configuration option "SynchronousZipStream" is set to "false", asynchronous transfers should be preferred for large amount of data, as the creation of the temporary file might lead to network timeouts.
- Transcode (character): If present, the DICOM files in the archive will be transcoded to the provided transfer syntax: https://orthanc.uclouvain.be/book/faq/transcoding.html
- UserData (list): In asynchronous mode, user data that will be attached to the job.

Returns: In asynchronous mode, information about the job that has been submitted to generate the archive: https://orthanc.uclouvain.be/book/users/advanced-rest.html#jobs.

Method `get_tools_default_encoding()`**:** Get default encoding

Get the default encoding that is used by Orthanc if parsing a DICOM instance without the `SpecificCharacterEncoding` tag, or during C-FIND. This corresponds to the configuration option `DefaultEncoding`.

Usage:

```
Orthanc$get_tools_default_encoding()
```

Returns: The name of the encoding.

Method `put_tools_default_encoding()`**:** Set default encoding

Change the default encoding that is used by Orthanc if parsing a DICOM instance without the `SpecificCharacterEncoding` tag, or during C-FIND. This corresponds to the configuration option `DefaultEncoding`.

Usage:

```
Orthanc$put_tools_default_encoding(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: The name of the encoding. Check out configuration option `DefaultEncoding` for the allowed values. (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_dicom_conformance()`**:** Get DICOM conformance

Get the DICOM conformance statement of Orthanc

Usage:

```
Orthanc$get_tools_dicom_conformance()
```

Returns: The DICOM conformance statement.

Method `post_tools_dicom_echo()`**:** Trigger C-ECHO SCU

Trigger C-ECHO SCU command against a DICOM modality described in the POST body, without having to register the modality in some `/modalities/{id}` (new in Orthanc 1.8.1)

Usage:

```
Orthanc$post_tools_dicom_echo(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- AET (character): AET of the remote DICOM modality
- CheckFind (logical): Issue a dummy C-FIND command after the C-GET SCU, in order to check whether the remote modality knows about Orthanc. This field defaults to the value of the `DicomEchoChecksFind` configuration option. New in Orthanc 1.8.1.
- Host (character): Host address of the remote DICOM modality (typically, an IP address)
- LocalAet (character): Whether to override the default DicomAet in the SCU connection initiated by Orthanc to this modality
- Manufacturer (character): Manufacturer of the remote DICOM modality (check configuration option `DicomModalities` for possible values
- Port (numeric): TCP port of the remote DICOM modality
- Timeout (numeric): Whether to override the default DicomScuTimeout in the SCU connection initiated by Orthanc to this modality
- UseDicomTls (logical): Whether to use DICOM TLS in the SCU connection initiated by Orthanc (new in Orthanc 1.9.0)

Returns: Nothing, invisibly.

Method `post_tools_execute_script()`**:** Execute Lua script

Execute the provided Lua script by the Orthanc server. This is very insecure for Orthanc servers that are remotely accessible. Since Orthanc 1.5.8, this route is disabled by default and can be enabled thanks to the `ExecuteLuaEnabled` configuration.

Usage:

```
Orthanc$post_tools_execute_script(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: The Lua script to be executed (text/plain).

Returns: Output of the Lua script.

Method `post_tools_find()`**:** Look for local resources

This URI can be used to perform a search on the content of the local Orthanc server, in a way that is similar to querying remote DICOM modalities using C-FIND SCU: https://orthanc.uclouvain.be/book/users/rest.html#perfo finds-within-orthanc

Usage:

```
Orthanc$post_tools_find(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- CaseSensitive (logical): Enable case-sensitive search for PN value representations (defaults to configuration option `CaseSensitivePN`)

- Expand (logical): If set to "true", retrieve detailed information about the individual resources, not only their Orthanc identifiers
- Full (logical): If set to `TRUE`, report the DICOM tags in full format (tags indexed by their hexadecimal format, associated with their symbolic name and their value)
- Labels (list): List of strings specifying which labels to look for in the resources (new in Orthanc 1.12.0)
- LabelsConstraint (character): Constraint on the labels, can be `All`, `Any`, or `None` (defaults to `All`, new in Orthanc 1.12.0)
- Level (character): Level of the query (`Patient`, `Study`, `Series` or `Instance`)
- Limit (numeric): Limit the number of reported resources
- MetadataQuery (list): Associative array containing the filter on the values of the metadata (new in Orthanc 1.12.5)
- OrderBy (list): Array of associative arrays containing the requested ordering (new in Orthanc 1.12.5)
- ParentPatient (character): Limit the reported resources to descendants of this patient (new in Orthanc 1.12.5)
- ParentSeries (character): Limit the reported resources to descendants of this series (new in Orthanc 1.12.5)
- ParentStudy (character): Limit the reported resources to descendants of this study (new in Orthanc 1.12.5)
- Query (list): Associative array containing the filter on the values of the DICOM tags
- RequestedTags (list): A list of DICOM tags to include in the response (applicable only if "Expand" is set to true). The tags requested tags are returned in the 'RequestedTags' field in the response. Note that, if you are requesting tags that are not listed in the Main Dicom Tags stored in DB, building the response might be slow since Orthanc will need to access the DICOM files. If not specified, Orthanc will return all Main Dicom Tags to keep backward compatibility with Orthanc prior to 1.11.0.
- ResponseContent (list): Defines the content of response for each returned resource. (this field, if present, overrides the "Expand" field). Allowed values are `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`, `Attachments`. If not specified, Orthanc will return `MainDicomTags`, `Metadata`, `Children`, `Parent`, `Labels`, `Status`, `IsStable`, `IsProtected`.(new in Orthanc 1.12.5)
- Short (logical): If set to `TRUE`, report the DICOM tags in hexadecimal format
- Since (numeric): Show only the resources since the provided index (in conjunction with `Limit`)

Returns: List containing either the Orthanc identifiers, or detailed information about the reported resources (if `Expand` argument is `TRUE`).

Method `get_tools_generate_uid()`**:** Generate an identifier

Generate a random DICOM identifier

Usage:

```
Orthanc$get_tools_generate_uid(level = NULL)
```

Arguments:

`level` (character) Type of DICOM resource among: `patient`, `study`, `series` or `instance`.

Returns: The generated identifier.

Method `post_tools_invalidate_tags()`**:** Invalidate DICOM-as-JSON summaries

Remove all the attachments of the type "DICOM-as-JSON" that are associated will all the DICOM instances stored in Orthanc. These summaries will be automatically re-created on the next access. This is notably useful after changes to the `Dictionary` configuration option. https://orthanc.uclouvain.be/book/faq/orthanc-storage.html#storage-area

Usage:

```
Orthanc$post_tools_invalidate_tags()
```

Returns: Nothing, invisibly.

Method `get_tools_labels()`**:** Get all the used labels

List all the labels that are associated with any resource of the Orthanc database

Usage:

```
Orthanc$get_tools_labels()
```

Returns: List containing the labels.

Method `get_tools_log_level()`**:** Get main log level

Get the main log level of Orthanc

Usage:

```
Orthanc$get_tools_log_level()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level()`**:** Set main log level

Set the main log level of Orthanc

Usage:

```
Orthanc$put_tools_log_level(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_dicom()`**:** Get log level for `dicom`

Get the log level of the log category `dicom`

Usage:

```
Orthanc$get_tools_log_level_dicom()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_dicom()`**:** Set log level for `dicom`

Set the log level of the log category `dicom`

Usage:

```
Orthanc$put_tools_log_level_dicom(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_generic()`**:** Get log level for `generic`
Get the log level of the log category `generic`

Usage:

```
Orthanc$get_tools_log_level_generic()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_generic()`**:** Set log level for `generic`
Set the log level of the log category `generic`

Usage:

```
Orthanc$put_tools_log_level_generic(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_http()`**:** Get log level for `http`
Get the log level of the log category `http`

Usage:

```
Orthanc$get_tools_log_level_http()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_http()`**:** Set log level for `http`
Set the log level of the log category `http`

Usage:

```
Orthanc$put_tools_log_level_http(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_jobs()`**:** Get log level for `jobs`
Get the log level of the log category `jobs`

Usage:

```
Orthanc$get_tools_log_level_jobs()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_jobs()`**:** Set log level for `jobs`
Set the log level of the log category `jobs`

Usage:

```
Orthanc$put_tools_log_level_jobs(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_lua()`**:** Get log level for `lua`
Get the log level of the log category `lua`

Usage:

```
Orthanc$get_tools_log_level_lua()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_lua()`**:** Set log level for `lua`
Set the log level of the log category `lua`

Usage:

```
Orthanc$put_tools_log_level_lua(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_plugins()`**:** Get log level for `plugins`
Get the log level of the log category `plugins`

Usage:

```
Orthanc$get_tools_log_level_plugins()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_plugins()`**:** Set log level for `plugins`
Set the log level of the log category `plugins`

Usage:

```
Orthanc$put_tools_log_level_plugins(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_log_level_sqlite()`**:** Get log level for `sqlite`
Get the log level of the log category `sqlite`

Usage:

```
Orthanc$get_tools_log_level_sqlite()
```

Returns: Possible values: `default`, `verbose` or `trace`.

Method `put_tools_log_level_sqlite()`**:** Set log level for `sqlite`

Set the log level of the log category `sqlite`

Usage:

```
Orthanc$put_tools_log_level_sqlite(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: Possible values: `default`, `verbose` or `trace` (text/plain).

Returns: Nothing, invisibly.

Method `post_tools_lookup()`**:** Look for DICOM identifiers

This URI can be used to convert one DICOM identifier to a list of matching Orthanc resources

Usage:

```
Orthanc$post_tools_lookup(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: The DICOM identifier of interest (i.e. the value of `PatientID`, `StudyInstanceUID`, `SeriesInstanceUID`, or `SOPInstanceUID`) (text/plain).

Returns: List containing a list of matching Orthanc resources, each item in the list corresponding to a JSON object with the fields `Type`, `ID` and `Path` identifying one DICOM resource that is stored by Orthanc.

Method `get_tools_metrics()`**:** Are metrics collected?

Returns a Boolean specifying whether Prometheus metrics are collected and exposed at `/tools/metrics-prometheus`

Usage:

```
Orthanc$get_tools_metrics()
```

Returns: `1` if metrics are collected, `0` if metrics are disabled.

Method `put_tools_metrics()`**:** Enable collection of metrics

Enable or disable the collection and publication of metrics at `/tools/metrics-prometheus`

Usage:

```
Orthanc$put_tools_metrics(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: `1` if metrics are collected, `0` if metrics are disabled (text/plain).

Returns: Nothing, invisibly.

Method `get_tools_metrics_prometheus()`**:** Get usage metrics

Get usage metrics of Orthanc in the Prometheus file format (OpenMetrics): https://orthanc.uclouvain.be/book/users/advance
rest.html#instrumentation-with-prometheus

Usage:

```
Orthanc$get_tools_metrics_prometheus()
```

Returns: Nothing, invisibly.

Method `get_tools_now()`**:** Get UTC time

Get UTC time

Usage:

```
Orthanc$get_tools_now()
```

Returns: The UTC time.

Method `get_tools_now_local()`**:** Get local time

Get local time

Usage:

```
Orthanc$get_tools_now_local()
```

Returns: The local time.

Method `post_tools_reconstruct()`**:** Reconstruct all the index

Reconstruct the index of all the tags of all the DICOM instances that are stored in Orthanc. This is notably useful after the deletion of resources whose children resources have inconsistent values with their sibling resources. Beware that this is a highly time-consuming operation, as all the DICOM instances will be parsed again, and as all the Orthanc index will be regenerated. If you have a large database to process, it is advised to use the Housekeeper plugin to perform this action resource by resource

Usage:

```
Orthanc$post_tools_reconstruct(json = NULL)
```

Arguments:

`json` (list) Named-list for request body. See Details.

Details: Request body JSON schema (application/json):

- ReconstructFiles (logical): Also reconstruct the files of the resources (e.g: apply Ingest-Transcoding, StorageCompression). 'false' by default. (New in Orthanc 1.11.0)

Returns: Nothing, invisibly.

Method `post_tools_reset()`**:** Restart Orthanc

Restart Orthanc

Usage:

```
Orthanc$post_tools_reset()
```

Returns: Nothing, invisibly.

Method `post_tools_shutdown()`**:** Shutdown Orthanc

Shutdown Orthanc

Usage:

```
Orthanc$post_tools_shutdown()
```

Returns: Nothing, invisibly.

Method `get_tools_unknown_sop_class_accepted()`**:** Is unknown SOP class accepted?

Shall Orthanc C-STORE SCP accept DICOM instances with an unknown SOP class UID?

Usage:

```
Orthanc$get_tools_unknown_sop_class_accepted()
```

Returns: `1` if accepted, `0` if not accepted.

Method `put_tools_unknown_sop_class_accepted()`**:** Set unknown SOP class accepted

Set whether Orthanc C-STORE SCP should accept DICOM instances with an unknown SOP class UID

Usage:

```
Orthanc$put_tools_unknown_sop_class_accepted(data = NULL)
```

Arguments:

`data` (bytes or character) Raw data for request body. See Details.

Details: Request body: `1` if accepted, `0` if not accepted (text/plain).

Returns: Nothing, invisibly.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Orthanc$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`OrthancAsync` *Asynchronous Orthanc API Client*

Description

Orthanc is an open-source, lightweight DICOM server for healthcare and medical research. This `R6` generator creates a client to access Orthanc's RESTful API. More details about the Orthanc REST API can be found here: https://orthanc.uclouvain.be/book/users/rest.html.

The full documentation of the Orthanc API can be found here: https://orthanc.uclouvain.be/api/.

Details

See Orthanc for more information.

Value

An `OrthancAsync` instance.

Super class

`orthanc::Orthanc` -> `OrthancAsync`

Methods

Public methods:

- `OrthancAsync$print()`
- `OrthancAsync$clone()`

Method `print()`**:** Print method for `OrthancAsync`.

Usage:

```
OrthancAsync$print(x, ...)
```

Arguments:

`x` Object to print.

`...` Further arguments passed to or from other methods.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
OrthancAsync$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Patient` *DICOM Patient Class*

Description

An abstract class for a DICOM Patient resource.

Value

An R6 instance of class `"Patient"`.

Super class

`orthanc::Resource` -> `Patient`

Active bindings

`patient_id` Patient ID

`name` Patient Name

`birth_date` Patient Birth Date

`sex` Patient Sex

`other_patient_ids` Other Patient IDs

`is_stable` Is stable?

`last_update` Last Update

`labels` Get or add labels

`protected` Get or set if patient is protected against recycling

`studies` Studies

`studies_ids` Studies identifiers

`series` Series

`series_ids` Series identifiers

`instances` Instances

`instances_ids` Instances identifiers

`instances_tags` Instances tags

`num_studies` Number of studies

`num_series` Number of series

`num_instances` Number of instances

`shared_tags` Shared tags

`statistics` Statistics

Methods

Public methods:

- `Patient$get_main_information()`
- `Patient$add_label()`
- `Patient$has_label()`
- `Patient$remove_label()`
- `Patient$get_zip_archive_content()`
- `Patient$download_archive()`
- `Patient$get_patient_module()`
- `Patient$anonymize()`
- `Patient$anonymize_as_job()`
- `Patient$modify()`
- `Patient$modify_as_job()`
- `Patient$get_shared_tags()`
- `Patient$remove_empty_studies()`
- `Patient$clone()`

Method `get_main_information()`**:** Get patient information.

Usage:

```
Patient$get_main_information()
```

Method `add_label()`**:** Add label to resource.

Usage:

```
Patient$add_label(label)
```

Arguments:

`label` Label.

Method `has_label()`**:** Test if resource has label.

Usage:

```
Patient$has_label(label)
```

Arguments:

`label` Label.

Method `remove_label()`**:** Delete label from resource.

Usage:

```
Patient$remove_label(label)
```

Arguments:

`label` Label.

Method `get_zip_archive_content()`**:** Get bytes of the zip archive.

Usage:

```
Patient$get_zip_archive_content()
```

Method `download_archive()`**:** Download zip archive to `path`.

Usage:

```
Patient$download_archive(path, stream = FALSE)
```

Arguments:

`path` Path on disk.

`stream` Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.

Method `get_patient_module()`**:** Get patient module in a simplified version

Usage:

```
Patient$get_patient_module()
```

Method `anonymize()`**:** Anonymize Patient

Usage:

```
Patient$anonymize(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.
`replace` Named-list of tags to replce.
`keep` List of tags to keep unchanged.
`keep_private_tags` Keep private tags from DICOM instance.
`keep_source` Keep original resource.
`priority` Priority of the job.
`permissive` Ignore errors during individual steps of the job?
`private_creator` Private creator to be used for private tags in replace.
`force` Force tags to be changed.
`dicom_version` Version of the DICOM standard to use for anonymization.

Method `anonymize_as_job()`**:** Anonymize Patient

Usage:

```
Patient$anonymize_as_job(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.
`replace` Named-list of tags to replce.
`keep` List of tags to keep unchanged.
`keep_private_tags` Keep private tags from DICOM instance.
`keep_source` Keep original resource.
`priority` Priority of the job.
`permissive` Ignore errors during individual steps of the job?
`private_creator` Private creator to be used for private tags in replace.
`force` Force tags to be changed.
`dicom_version` Version of the DICOM standard to use for anonymization.

Method `modify()`**:** Modify Patient

Usage:

```
Patient$modify(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
```

```
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `modify_as_job()`**:** Modify Patient

Usage:

```
Patient$modify_as_job(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `get_shared_tags()`**:** Retrieve the shared tags of the patient.

Usage:

```
Patient$get_shared_tags()
```

Method `remove_empty_studies()`**:** Remove empty studies from patient.

Usage:

```
Patient$remove_empty_studies()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Patient$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`query_orthanc` *Query data in the Orthanc server*

Description

Query data in the Orthanc server

Usage

```
query_orthanc(
  client,
  level,
  query = list(),
  labels = character(),
  labels_constraint = "All",
  limit = 1000L,
  since = 0L,
  retrieve_all_resources = TRUE,
  lock_children = FALSE
)
```

Arguments

`client`	Orthanc API client.
`level`	Level of the query ('Patient', 'Study', 'Series', 'Instance').
`query`	Named-list that specifies the filters on the level related DICOM tags.
`labels`	Character vector of labels to look for in resources.
`labels_constraint`	Contraint on the labels ('All', 'Any', 'None').
`limit`	Limit the number of reported instances (default is `1000L`).
`since`	Show only the resources since the index specified in the `since` parameter.
`retrieve_all_resources`	Retrieve all resources since the index specified in the `since` parameter.
`lock_children`	If `lock_children` is `TRUE`, the resource children (e.g., instances of a series via `Series.instances`) will be cached at the first query rather than queried every time. This is useful when you want to filter the children of a resource and want to maintain the filter result.

Value

A `list` of resources.

Examples

```
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")
query_orthanc(client, "Patient", list(PatientName = "HN_P001"))
```

`RemoteModality` *Remote Modality Class*

Description

Wrapper around Orthanc API when dealing with a remote modality

Value

An instance of `RemoteModality`.

Super class

`orthanc::Modality` -> `RemoteModality`

`Resource` *DICOM Resource Class*

Description

An abstract class for a DICOM Resource

Value

An R6 instance of class `"Resource"`.

Active bindings

`identifier` Orthanc identifier of the resource.

`main_dicom_tags` Main DICOM tags for the resource.

`is_locked` Get or set whether resource is locked.

Methods

Public methods:

- `Resource$new()`
- `Resource$get_main_information()`
- `Resource$print()`
- `Resource$set_child_resources()`
- `Resource$clone()`

Method `new()`**:** Create a new Resource.

Usage:

```
Resource$new(id, client, lock_children = FALSE)
```

Arguments:

`id` Orthanc identifier of the resource.

`client` Orthanc client.

`lock_children` If `lock_children` is `TRUE`, the resource children (e.g., instances of a series via `Series$instances`) will be cached at the first query rather than queried every time. This is useful when you want to filter the children of a resource and want to maintain the filter result.

Method `get_main_information()`**:** Get main information for the resource.

Usage:

```
Resource$get_main_information()
```

Method `print()`**:** Print a `Resource` object.

Usage:

```
Resource$print(...)
```

Arguments:

`...` Not currently used.

Method `set_child_resources()`**:** Set child resources.

Usage:

```
Resource$set_child_resources(resources)
```

Arguments:

`resources` List of resources.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Resource$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`retrieve_and_write_nifti` *Retrieve and write resources to a path*

Description

Retrieve the NIfTI file or ZIP Archive contents for a list of resources and write them to a `path` on disk. If `mirai::daemons()` has been used to set persistent background processes, these functions will write resources to disk in parallel using all available processes.

Usage

```
retrieve_and_write_nifti(
  resources,
  path,
  compress = TRUE,
  stream = FALSE,
  progress = rlang::is_interactive()
)

retrieve_and_write_archives(
  resources,
  path,
  stream = FALSE,
  progress = rlang::is_interactive()
)

retrieve_and_write_images(resources, path, progress = rlang::is_interactive())
```

Arguments

`resources`	List of resources.
`path`	Path where you want to write the resources.
`compress`	Compress to gzip (`nii.gz`) Default is `TRUE`.
`stream`	Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.
`progress`	Whether to show progress bars. By default, progress bars are enabled in interactive sessions (i.e., if `rlang::is_interactive()` returns `TRUE`).

Value

Nothing, invisibly.

Examples

```
## Not run:
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")
study_id <- client$get_studies()[[1]]
study <- Study$new(study_id, client)

retrieve_and_write_nifti(study$series, tempdir())

retrieve_and_write_archives(study$series, tempdir())

retrieve_and_write_images(study$instances, tempdir())

## End(Not run)
```

`retrieve_and_write_patients` *Retrieve and write patients to a path*

Description

Retrieve the DICOM file contents for a list of Patients and write them to a `path` on disk. DICOM files are saved to disk in a directory structure of `Patient` -> `Study` -> `Series` -> `File`. If `mirai::daemons()` has been used to set persistent background processes, this function will write patients to disk in parallel using all available processes.

Usage

```
retrieve_and_write_patients(
  patients,
  path,
  stream = FALSE,
  progress = rlang::is_interactive()
)
```

Arguments

`patients`	List of Patients.
`path`	Path where you want to write the patients (files).
`stream`	Should the resources be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.
`progress`	Whether to show progress bars. By default, progress bars are enabled in interactive sessions (i.e., if `rlang::is_interactive()` returns `TRUE`).

Value

Nothing, invisibly.

Examples

```
## Not run:
client <- Orthanc$new("https://orthanc.uclouvain.be/demo")

patients <- find_patients(client, query = list(PatientName = "HN_P001"))

retrieve_and_write_patients(patients, tempdir())

## End(Not run)
```

`Series` *DICOM Series Class*

Description

An abstract class for a DICOM Series resource.

Value

An R6 instance of class `"Series"`.

Super class

`orthanc::Resource` -> `Series`

Active bindings

`instances` Instances

`instances_ids` Instances identifiers

`num_instances` Number of instances

`instances_tags` Instances tags

`uid` SeriesInstanceUID

`manufacturer` Manufacturer

`date` Series Date

`modality` Modality

`series_number` Series Number

`performed_procedure_step_description` Performed Procedure Step Description

`protocol_name` Protocol Name

`station_name` Station Name

`description` Series Description

`body_part_examined` Body Part Examined

`sequence_name` Sequence Name

`cardiac_number_of_images` Cardiac Number of Images

`image_in_acquisition` Images in Acquisition

`number_of_temporal_positions` Number of Temporal Positions

`number_of_slices` Number of Slices

`number_of_time_slices` Number of Time Slices

`image_orientation_patient` Image Orientation Patient

`series_type` Series Type

`operators_name` Operators Name

`acquisition_device_processing_description` Acquisition Device Processing Description

`contrast_bolus_agent` Contrast Bolus Agent

`is_stable` Is stable?

`last_update` Last Update

`labels` Get or add labels

`study_identifier` Parent study identifier

`parent_study` Parent study

`parent_patient` Parent patient

`shared_tags` Shared tags

`statistics` Statistics

Methods

Public methods:

- `Series$get_main_information()`
- `Series$add_label()`
- `Series$has_label()`
- `Series$remove_label()`
- `Series$anonymize()`
- `Series$anonymize_as_job()`
- `Series$modify()`
- `Series$modify_as_job()`
- `Series$get_zip_archive_content()`
- `Series$download_archive()`
- `Series$get_shared_tags()`
- `Series$remove_empty_instances()`
- `Series$get_nifti_file_content()`
- `Series$download_nifti()`
- `Series$clone()`

Method `get_main_information()`**:** Get series information.

Usage:

```
Series$get_main_information()
```

Method `add_label()`**:** Add label to resource.

Usage:

```
Series$add_label(label)
```

Arguments:

`label` Label.

Method `has_label()`**:** Test if resource has label.

Usage:

```
Series$has_label(label)
```

Arguments:

`label` Label.

Method `remove_label()`**:** Delete label from resource.

Usage:

```
Series$remove_label(label)
```

Arguments:

`label` Label.

Method `anonymize()`**:** Anonymize Series

Usage:

```
Series$anonymize(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`keep_private_tags` Keep private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

`dicom_version` Version of the DICOM standard to use for anonymization.

Method `anonymize_as_job()`**:** Anonymize Series

Usage:

```
Series$anonymize_as_job(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`keep_private_tags` Keep private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

`dicom_version` Version of the DICOM standard to use for anonymization.

Method `modify()`**:** Modify Series

Usage:

```
Series$modify(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `modify_as_job()`**:** Modify Series

Usage:

```
Series$modify_as_job(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `get_zip_archive_content()`**:** Get bytes of the zip archive.

Usage:

```
Series$get_zip_archive_content()
```

Method `download_archive()`**:** Download zip archive to `path`.

Usage:

```
Series$download_archive(path, stream = FALSE)
```

Arguments:

`path` Path on disk.

`stream` Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.

Method `get_shared_tags()`**:** Retrieve the shared tags of the series.

Usage:

```
Series$get_shared_tags()
```

Method `remove_empty_instances()`: Remove empty instances from series.

Usage:

```
Series$remove_empty_instances()
```

Method `get_nifti_file_content()`: Get bytes of NIfTI file content.

Usage:

```
Series$get_nifti_file_content(compress = TRUE)
```

Arguments:

`compress` Compress to gzip (nii.gz) Default is `TRUE`.

Method `download_nifti()`: Download series as NIfTI.

Usage:

```
Series$download_nifti(path, compress = TRUE, stream = FALSE)
```

Arguments:

`path` Path on disk.

`compress` Compress to gzip (`nii.gz`) Default is `TRUE`.

`stream` Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource contents are retrieved in their entirety and written to disk all at once.

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
Series$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`Study` *DICOM Study Class*

Description

An abstract class for a DICOM Study resource.

Value

An R6 instance of class `"Study"`.

Super class

`orthanc::Resource` -> `Study`

Active bindings

`patient_identifier` Parent patient identifier

`parent_patient` Parent patient

`referring_physician_name` Referring Physician Name

`requesting_physician` Requesting Physician

`date` Study Date

`study_id` Study ID

`uid` StudyInstanceUID

`patient_information` Patient Main DICOM Tags

`series` Series

`series_ids` Series identifiers

`instances` Instances

`instances_ids` Instances identifiers

`num_series` Number of series

`num_instances` Number of instances

`instances_tags` Instances tags

`accession_number` Accession Number

`description` Description

`institution_name` Institution Name

`requested_procedure_description` Requested Procedure Description.

`is_stable` Is stable?

`last_update` Last Update

`labels` Get or add labels

`shared_tags` Shared Tags

`statistics` Statistics

Methods

Public methods:

- `Study$get_main_information()`
- `Study$add_label()`
- `Study$has_label()`
- `Study$remove_label()`
- `Study$anonymize()`
- `Study$anonymize_as_job()`
- `Study$modify()`
- `Study$modify_as_job()`
- `Study$get_zip_archive_content()`
- `Study$download_archive()`

- Study$get_shared_tags()
- Study$remove_empty_series()
- Study$clone()

Method `get_main_information()`**:** Get study information.

Usage:

```
Study$get_main_information()
```

Method `add_label()`**:** Add label to resource.

Usage:

```
Study$add_label(label)
```

Arguments:

`label` Label.

Method `has_label()`**:** Test if resource has label.

Usage:

```
Study$has_label(label)
```

Arguments:

`label` Label.

Method `remove_label()`**:** Delete label from resource.

Usage:

```
Study$remove_label(label)
```

Arguments:

`label` Label.

Method `anonymize()`**:** Anonymize Study

Usage:

```
Study$anonymize(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`keep_private_tags` Keep private tags from DICOM instance.
`keep_source` Keep original resource.
`priority` Priority of the job.
`permissive` Ignore errors during individual steps of the job?
`private_creator` Private creator to be used for private tags in replace.
`force` Force tags to be changed.
`dicom_version` Version of the DICOM standard to use for anonymization.

Method `anonymize_as_job()`**:** Anonymize Study

Usage:

```
Study$anonymize_as_job(
  remove = list(),
  replace = list(),
  keep = list(),
  keep_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE,
  dicom_version = NULL
)
```

Arguments:

`remove` List of tags to remove.
`replace` Named-list of tags to replce.
`keep` List of tags to keep unchanged.
`keep_private_tags` Keep private tags from DICOM instance.
`keep_source` Keep original resource.
`priority` Priority of the job.
`permissive` Ignore errors during individual steps of the job?
`private_creator` Private creator to be used for private tags in replace.
`force` Force tags to be changed.
`dicom_version` Version of the DICOM standard to use for anonymization.

Method `modify()`**:** Modify Study

Usage:

```
Study$modify(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `modify_as_job()`**:** Modify Study as Job

Usage:

```
Study$modify_as_job(
  remove = list(),
  replace = list(),
  keep = list(),
  remove_private_tags = FALSE,
  keep_source = TRUE,
  priority = 0L,
  permissive = FALSE,
  private_creator = NULL,
  force = FALSE
)
```

Arguments:

`remove` List of tags to remove.

`replace` Named-list of tags to replce.

`keep` List of tags to keep unchanged.

`remove_private_tags` Remove private tags from DICOM instance.

`keep_source` Keep original resource.

`priority` Priority of the job.

`permissive` Ignore errors during individual steps of the job?

`private_creator` Private creator to be used for private tags in replace.

`force` Force tags to be changed.

Method `get_zip_archive_content()`**:** Get bytes of the zip archive.

Usage:

```
Study$get_zip_archive_content()
```

Method `download_archive()`**:** Download zip archive to `path`.

Usage:

```
Study$download_archive(path, stream = FALSE)
```

Arguments:

`path` Path on disk.

`stream` Should the resource be streamed and written to disk in chunks? Default is `FALSE`, which means the resource file contents are retrieved in their entirety and written to disk all at once.

Method `get_shared_tags()`**:** Retrieve the shared tags of the study.

Usage:

```
Study$get_shared_tags()
```

Method `remove_empty_series()`**:** Remove empty series from study.

Usage:

```
Study$remove_empty_series()
```

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Study$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Index