Package: datasetjson (via r-universe)

October 22, 2026

Type Package

Title Read and Write CDISC Dataset JSON Files

Version 0.4.1

Description Read, construct and write CDISC (Clinical Data Interchange Standards Consortium) Dataset JSON (JavaScript Object Notation) files, while validating per the Dataset JSON schema file, as described in CDISC (2023) <https://www.cdisc.org/standards/data-exchange/dataset-json>.

URL https://atorus-research.github.io/datasetjson/

BugReports https://github.com/atorus-research/datasetjson/issues/

Encoding UTF-8

Language en-US

License Apache License (>= 2)

Copyright The included 'yyjson' C library is Copyright (c) 2020 YaoYuan and is released under the MIT license. See inst/COPYRIGHTS.

LazyData true

Depends R (>= 4.0)

Imports hms

Suggests testthat (>= 3.1.7), jsonvalidate (>= 1.3.1), jsonlite (>= 1.8.0), mockery, knitr, haven, rmarkdown, withr, purrr, tibble, dplyr, lubridate, data.table

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation yes

Author Mike Stackhouse [aut, cre] (ORCID: <https://orcid.org/0000-0001-6030-723X>), Nicholas Masel [aut], Atorus Research, Inc. [cph], Yao Yuan [cph] (Author of the bundled yyjson C library)

Maintainer Mike Stackhouse <mike.stackhouse@atorusresearch.com>

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

Date/Publication 2026-10-22 19:21:31 UTC

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

RemoteRef HEAD

RemoteSha 3f986ae8eaeb8b9b5b3a2a302d97e6198a76a6ea

Contents

dataset_json 2
datasetjson_example 5
get_column_metadata 5
iris_items 6
read_dataset_dsjc 7
read_dataset_json 7
read_dataset_ndjson 9
schema_1_1_0 10
schema_ndjson_1_1_0 10
set_source_system 11
set_variable_attributes 12
validate_dataset_dsjc 13
validate_dataset_json 14
validate_dataset_ndjson 15
write_dataset_dsjc 15
write_dataset_json 17
write_dataset_ndjson 18

Index **20**

`dataset_json` *Create a Dataset JSON Object*

Description

Create the base object used to write a Dataset JSON file.

Usage

```
dataset_json(
  .data,
  file_oid = NULL,
  last_modified = NULL,
  originator = NULL,
  sys = NULL,
  sys_version = NULL,
  study = NULL,
```

```
  metadata_version = NULL,
  metadata_ref = NULL,
  item_oid = NULL,
  name = NULL,
  dataset_label = NULL,
  columns = NULL,
  version = "1.1.0"
)
```

Arguments

`.data`	Input data to contain within the Dataset JSON file. Written to the itemData parameter.
`file_oid`	fileOID parameter, defined as "A unique identifier for this file." (optional)
`last_modified`	The date/time the source database was last modified before creating the Dataset-JSON file (optional)
`originator`	originator parameter, defined as "The organization that generated the Dataset-JSON file." (optional)
`sys`	sourceSystem.name parameter, defined as "The computer system or database management system that is the source of the information in this file." (Optional, required if coupled with sys_version)
`sys_version`	sourceSystem.Version, defined as "The version of the sourceSystem" (Optional, required if coupled with sys)
`study`	Study OID value (optional)
`metadata_version`	Metadata version OID value (optional)
`metadata_ref`	Metadata reference (i.e. path to Define.xml) (optional)
`item_oid`	ID used to label dataset with the itemGroupData parameter. Defined as "Object of Datasets. Key value is a unique identifier for Dataset, corresponding to ItemGroupDef/@OID in Define-XML."
`name`	Dataset name
`dataset_label`	Dataset Label
`columns`	Variable level metadata for the Dataset JSON object. See details for format requirements.
`version`	The DatasetJSON version to use. Currently only 1.1.0 is supported.

Details

The `columns` parameter should be provided as a dataframe based off the Dataset JSON Specification:

- **itemOID**: *string, required*: Unique identifier for the variable that may also function as a foreign key to an ItemDef/@OID in an associated Define-XML file. See the ODM specification for OID considerations.
- **name**: *string, required*: Variable name

- **label**: *string, required*: Variable label
- **dataType**: *string, required*: Logical data type of the variable. The dataType attribute represents the planned specificity of the data. See the ODM Data Formats specification for details. -**targetDataType**: *string, optional*: Indicates the data type into which the receiving system must transform the associated Dataset-JSON variable. The variable with the data type attribute of dataType must be converted into the targetDataType when transforming the Dataset-JSON dataset into a format for operational use (e.g., SAS dataset, R dataframe, loading into a system's data store). Only specify targetDataType when it is different from the dataType attribute or the JSON data type and the data needs to be transformed by the receiving system. See the Supported Column Data Type Combinations table for details on usage. See the User's Guide for additional information.
- **length**: *integer, optional*: Specifies the number of characters allowed for the variable value when it is represented as a text.
- **displayFormat**: *string, optional: A SAS display format value used for data visualization of numeric float and date values.
- **keySequence**: *integer, optional*: Indicates that this item is a key variable in the dataset structure. It also provides an ordering for the keys.

Note that DatasetJSON is on version 1.1.0. Based off findings from the pilot, version 1.1.0 reflects feedback from the user community. Support for 1.0.0 has been deprecated.

Value

dataset_json object pertaining to the specific Dataset JSON version specific

Examples

```
# Create a basic object
ds_json <- dataset_json(
  iris,
  file_oid = "/some/path",
  last_modified = "2023-02-15T10:23:15",
  originator = "Some Org",
  sys = "source system",
  sys_version = "1.0",
  study = "SOMESTUDY",
  metadata_version = "MDV.MSGv2.0.SDTMIG.3.3.SDTM.1.7",
  metadata_ref = "some/define.xml",
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)

# Attach attributes directly
ds_json <- dataset_json(iris, columns = iris_items)
ds_json <- set_file_oid(ds_json, "/some/path")
ds_json <- set_last_modified(ds_json, "2025-01-21T13:34:50")
ds_json <- set_originator(ds_json, "Some Org")
ds_json <- set_source_system(ds_json, "source system", "1.0")
```

```
ds_json <- set_study_oid(ds_json, "SOMESTUDY")
ds_json <- set_metadata_ref(ds_json, "some/define.xml")
ds_json <- set_metadata_version(ds_json, "MDV.MSGv2.0.SDTMIG.3.3.SDTM.1.7")
ds_json <- set_item_oid(ds_json, "IG.IRIS")
ds_json <- set_dataset_name(ds_json, "Iris")
ds_json <- set_dataset_label(ds_json, "The Iris Dataset")
```

---

datasetjson_example *Get path to a datasetjson example file*

Description

datasetjson comes bundled with sample files in its `inst/extdata` directory. This function makes them easy to access.

Usage

```
datasetjson_example(file = NULL)
```

Arguments

`file`	Name of file. If `NULL`, the example files will be listed.

Value

A file path string, or a character vector of available files if `file` is `NULL`.

Examples

```
datasetjson_example()
datasetjson_example("dm.json")
```

---

get_column_metadata *Extract column metadata to data frame*

Description

This function pulls out the column metadata from the `datasetjson` object attributes into a more user-friendly data.frame.

Usage

```
get_column_metadata(x)
```

Arguments

`x`	A datasetjson object

Value

A data frame containing the columns metadata

Examples

```
ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)

get_column_metadata(ds_json)
```

---

iris_items *Example Variable Metadata for Iris*

Description

Example of the necessary variable metadata included in a Dataset JSON file based on the Iris data frame.

Usage

```
iris_items
```

Format

`iris_items` **A data frame with 5 rows and 6 columns::**

itemOID Unique identifier for Variable. Must correspond to ItemDef/@OID in Define-XML.

name Display format supports data visualization of numeric float and date values.

label Label for Variable

dataType Data type for Variable

length Length for Variable

keySequence Indicates that this item is a key variable in the dataset structure. It also provides an ordering for the keys.

read_dataset_dsjc *Read a Dataset JSON Compressed (DSJC) file to a datasetjson object*

Description

Reads the compressed representation of Dataset JSON: a zLib stream holding Dataset NDJSON content. The object returned is the same one `read_dataset_json()` and `read_dataset_ndjson()` return for the equivalent uncompressed file, metadata attributes included.

Usage

```
read_dataset_dsjc(file)
```

Arguments

`file`	File path or URL of a DSJC file, or a raw vector holding the compressed bytes

Value

A dataframe with additional attributes attached containing the DatasetJSON metadata.

Examples

```
# Read one of the example files shipped with the package
dm <- read_dataset_dsjc(datasetjson_example("dm.dsjc"))

# Or from bytes held in memory
ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
dat <- read_dataset_dsjc(write_dataset_dsjc(ds_json))
```

read_dataset_json *Read a Dataset JSON to datasetjson object*

Description

This function validates a dataset JSON file against the Dataset JSON schema, and if valid returns a datasetjson object. The Dataset JSON file can be either a file path on disk of a URL which contains the Dataset JSON file.

Usage

```
read_dataset_json(file)
```

Arguments

file	File path or URL of a Dataset JSON file

Details

The resulting dataframe contains the additional metadata available on the Dataset JSON file within the attributes to make this accessible to the user. Note that these attributes are only populated if available.

- **sourceSystem**: The information system from which the content of this dataset was source, including system name and version.
- **datasetJSONVersion**: The version of the Dataset-JSON standard used to create the dataset.
- **fileOID**: A unique identifier for this dataset.
- **dbLastModifiedDateTime**: The date/time the source database was last modified before creating the Dataset-JSON file.
- **originator**: The organization that generated the Dataset-JSON dataset.
- **studyOID**: Unique identifier for the study that may also function as a foreign key to a Study/@OID in an associated Define-XML document, or to any studyOID references that are used as keys in other documents;
- **metaDataVersionOID**: Unique identifier for the metadata version that may also function as a foreign key to a MetaDataVersion/@OID in an associated Define-XML file
- **metaDataRef**: URI for the metadata file describing the dataset (e.g., a Define-XML file).
- **itemGroupOID**: Unique identifier for the dataset that may also function as a foreign key to an ItemGroupDef/@OID in an associated Define-XML file.
- **name**: The human-readable name for the dataset.
- **label**: A short description of the dataset.
- **columns**: An array of metadata objects that describe the dataset variables. See `dataset_json()` for further information on the contents of these fields.

Value

A dataframe with additional attributes attached containing the DatasetJSON metadata.

Examples

```
# Read from disk
dat <- read_dataset_json(datasetjson_example("dm.json"))

# Read from a URL
## Not run:
  dat <- read_dataset_json('https://www.somesite.com/file.json')
```

```
## End(Not run)

# Read from an already imported character vector
ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
js <- write_dataset_json(ds_json)
dat <- read_dataset_json(js)
```

`read_dataset_ndjson` *Read a Dataset NDJSON file to a datasetjson object*

Description

Reads a newline-delimited JSON (NDJSON) file following the Dataset JSON v1.1.0 specification. Line 1 of the file holds the dataset metadata and each subsequent line holds one data row as a JSON array.

Usage

```
read_dataset_ndjson(file)
```

Arguments

`file`	File path or URL of a Dataset NDJSON file, or a character string containing the NDJSON content itself

Details

NDJSON and JSON representations of Dataset JSON carry the same content, so the object returned here is the same as the one `read_dataset_json()` returns for the equivalent `.json` file, including all of the metadata attached as attributes. See `read_dataset_json()` for the full list.

Value

A dataframe with additional attributes attached containing the DatasetJSON metadata.

Examples

```
# Read one of the example files shipped with the package
dm <- read_dataset_ndjson(datasetjson_example("dm.ndjson"))

# Or from NDJSON text held in memory
ds_json <- dataset_json(
  iris,
```

```
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
dat <- read_dataset_ndjson(write_dataset_ndjson(ds_json))
```

---

schema_1_1_0 *Dataset JSON Schema Version 1.1.0*

Description

This object is a character vector holding the schema for Dataset JSON Version 1.1.0

Usage

```
schema_1_1_0
```

Format

`schema_1_1_0`:
A character vector with 1 element

---

schema_ndjson_1_1_0 *Dataset NDJSON Schema Version 1.1.0*

Description

This object is a character vector holding the schema used to validate the metadata line of a Dataset NDJSON file. It is the CDISC-published schema with three definitions hoisted into `$defs` so that the references to them resolve; see `data-raw/data.R`.

Usage

```
schema_ndjson_1_1_0
```

Format

`schema_ndjson_1_1_0`:
A character vector with 1 element

`set_source_system` *Dataset Metadata Setters*

Description

Set information about the file, source system, study, and dataset used to generate the Dataset JSON object.

Usage

```
set_source_system(x, sys, sys_version)

set_originator(x, originator)

set_file_oid(x, file_oid)

set_study_oid(x, study)

set_metadata_version(x, metadata_version)

set_metadata_ref(x, metadata_ref)

set_item_oid(x, item_oid)

set_dataset_name(x, name)

set_dataset_label(x, dataset_label)

set_last_modified(x, last_modified)
```

Arguments

`x`	datasetjson object
`sys`	sourceSystem.name parameter, defined as "The computer system or database management system that is the source of the information in this file." (Optional, required if coupled with sys_version)
`sys_version`	sourceSystem.Version, defined as "The version of the sourceSystem" (Optional, required if coupled with sys)
`originator`	originator parameter, defined as "The organization that generated the Dataset-JSON file." (optional)
`file_oid`	fileOID parameter, defined as "A unique identifier for this file." (optional)
`study`	Study OID value (optional)
`metadata_version`	Metadata version OID value (optional)
`metadata_ref`	Metadata reference (i.e. path to Define.xml) (optional)

item_oid	ID used to label dataset with the itemGroupData parameter. Defined as "Object of Datasets. Key value is a unique identifier for Dataset, corresponding to ItemGroupDef/@OID in Define-XML."
name	Dataset name
dataset_label	Dataset Label
last_modified	The date/time the source database was last modified before creating the Dataset-JSON file (optional)

Details

The fileOID parameter should be structured following description outlined in the ODM V2.0 specification. "FileOIDs should be universally unique if at all possible. One way to ensure this is to prefix every FileOID with an internet domain name owned by the creator of the ODM file or database (followed by a forward slash, "/"). For example, FileOID="BestPharmaceuticals.com/Study5894/1" might be a good way to denote the first file in a series for study 5894 from Best Pharmaceuticals."

Value

datasetjson object

Examples

```
ds_json <- dataset_json(iris, columns = iris_items)
ds_json <- set_file_oid(ds_json, "/some/path")
ds_json <- set_last_modified(ds_json, "2025-01-21T13:34:50")
ds_json <- set_originator(ds_json, "Some Org")
ds_json <- set_source_system(ds_json, "source system", "1.0")
ds_json <- set_study_oid(ds_json, "SOMESTUDY")
ds_json <- set_metadata_ref(ds_json, "some/define.xml")
ds_json <- set_metadata_version(ds_json, "MDV.MSGv2.0.SDTMIG.3.3.SDTM.1.7")
ds_json <- set_item_oid(ds_json, "IG.IRIS")
ds_json <- set_dataset_name(ds_json, "Iris")
ds_json <- set_dataset_label(ds_json, "The Iris Dataset")
```

---

set_variable_attributes

Assign Dataset JSON attributes to data frame columns

Description

Using the columns element of the Dataset JSON file, assign the available metadata to individual columns

Usage

```
set_variable_attributes(x)
```

Arguments

x	A datasetjson object

Value

A datasetjson object with attributes assigned to individual variables

Examples

```
ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)

ds_json <- set_variable_attributes(ds_json)
```

---

validate_dataset_dsjc *Validate a Dataset JSON Compressed (DSJC) file*

Description

Decompresses the zLib stream and validates the Dataset NDJSON content it holds, as `validate_dataset_ndjson()` does: the metadata object on line 1 is checked against the Dataset NDJSON v1.1.0 schema, and each subsequent line must be a JSON array carrying one value per declared column.

Usage

```
validate_dataset_dsjc(x)
```

Arguments

x	File path or URL of a DSJC file, or a raw vector holding the compressed bytes

Value

A data frame of errors, empty when the file is valid

Examples

```
validate_dataset_dsjc(datasetjson_example("dm.dsjc"))
```

`validate_dataset_json` *Validate a Dataset JSON file*

Description

This function calls `jsonvalidate::json_validate()` directly, with the parameters necessary to retrieve the error information of an invalid JSON file per the Dataset JSON schema.

Usage

```
validate_dataset_json(x)
```

Arguments

`x`	File path or URL of a Dataset JSON file, or a character vector holding JSON text

Value

A data frame

Examples

```
# Validate a file on disk
validate_dataset_json(datasetjson_example("dm.json"))

# Validate from a URL
## Not run:
  validate_dataset_json('https://www.somesite.com/file.json')


ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
js <- write_dataset_json(ds_json)

validate_dataset_json(js)

## End(Not run)
```

`validate_dataset_ndjson` *Validate a Dataset NDJSON file*

Description

Checks a Dataset NDJSON file in two parts, matching how the format is structured: the metadata object on line 1 is validated against the Dataset NDJSON v1.1.0 schema, and each subsequent line is checked to be a JSON array carrying one value per declared column.

Usage

```
validate_dataset_ndjson(x)
```

Arguments

`x`	File path or URL of a Dataset NDJSON file, or a character vector holding the NDJSON text

Value

A data frame of errors, empty when the file is valid

Examples

```
# Validate a file on disk
validate_dataset_ndjson(datasetjson_example("dm.ndjson"))

ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)

validate_dataset_ndjson(write_dataset_ndjson(ds_json))
```

`write_dataset_dsjc` *Write out a Dataset JSON Compressed (DSJC) file*

Description

Writes the compressed representation of Dataset JSON: the Dataset NDJSON content of the dataset, compressed as a zLib stream. The format carries no wrapper of its own - the file is the zLib stream and nothing else - and uses the `.dsjc` extension.

Usage

```
write_dataset_dsjc(
  x,
  file,
  float_as_decimals = FALSE,
  level = 9L,
  digits = NULL
)
```

Arguments

x	datasetjson object
file	File path to save the DSJC file. If not provided, the compressed bytes are returned as a raw vector.
float_as_decimals	If TRUE, write float variables as "decimal" data types. This is an interoperability choice; it is not needed for precision, as numbers are written at full precision either way, and setting it raises a warning to that effect.
level	zLib compression level, 0 (none) to 9 (maximum). Defaults to 9, which the specification recommends for data exchange. Lower levels compress faster and less.
digits	Deprecated and ignored. See `write_dataset_json()`.

Details

Rows are compressed as they are serialized, so writing a large dataset never holds its uncompressed NDJSON in memory.

The default `level = 9` follows the specification's recommendation for data exchange. Note that the top of the range buys little: on a 26 MB dataset, level 1 wrote in a fifth of the time for a file only 4% larger. If write time matters more than the last few percent, a lower level is a reasonable choice. Read time is unaffected by the level used to write.

Value

NULL when writing to a file, otherwise a raw vector

Examples

```
ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
bytes <- write_dataset_dsjc(ds_json)

# Write to disk
write_dataset_dsjc(ds_json, tempfile(fileext = ".dsjc"))
```

`write_dataset_json` *Write out a Dataset JSON file*

Description

Write out a Dataset JSON file

Usage

```
write_dataset_json(
  x,
  file,
  pretty = FALSE,
  float_as_decimals = FALSE,
  digits = NULL
)
```

Arguments

`x`	datasetjson object
`file`	File path to save Dataset JSON file
`pretty`	If TRUE, write with readable formatting. *Note: The Dataset JSON standard prefers compressed formatting without line feeds. It is not recommended you use pretty printing for submission purposes.*
`float_as_decimals`	If TRUE, write float variables as the "decimal" data type, quoting the numbers as JSON strings rather than writing them as JSON numbers. This is an interoperability choice for systems that expect the decimal type; it is not needed for precision, as numbers are written at full precision either way, and setting it raises a warning to that effect. See the Dataset JSON user guide for more information. Defaults to FALSE
`digits`	Deprecated and ignored. Decimals are written at whatever precision reads back as the same value, so there is no precision for this argument to control. It is ignored, and supplying it warns. If you need values rendered at a fixed precision, format the column to character yourself and declare it as `decimal/decimal` in the column metadata; the writer passes such columns through verbatim.

Value

NULL when file written to disk, otherwise character string

Examples

```
# Write to character object
ds_json <- dataset_json(
  iris,
```

```
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
js <- write_dataset_json(ds_json)

# Write to disk
write_dataset_json(ds_json, tempfile(fileext = ".json"))

# float_as_decimals writes floats as the "decimal" type, quoting the numbers.
# It is an interoperability choice for systems that require that type - it is
# not needed for precision, and setting it warns to say so.
js <- suppressWarnings(write_dataset_json(ds_json, float_as_decimals = TRUE))

# `digits` is deprecated and ignored; decimals are written at whatever
# precision reads back as the same value
js <- suppressWarnings(write_dataset_json(ds_json, digits = 16))
```

`write_dataset_ndjson` *Write out a Dataset NDJSON file*

Description

Writes the newline-delimited JSON representation of Dataset JSON: the dataset metadata as a single JSON object on line 1, then one JSON array per data row. The content is identical to what `write_dataset_json()` produces; only the framing differs, which is what makes NDJSON straightforward to stream a row at a time.

Usage

```
write_dataset_ndjson(x, file, float_as_decimals = FALSE, digits = NULL)
```

Arguments

`x`	datasetjson object
`file`	File path to save the Dataset NDJSON file. If not provided, the NDJSON is returned as a character string.
`float_as_decimals`	If TRUE, write float variables as "decimal" data types, serialized as JSON strings rather than numbers. This is an interoperability choice; it is not needed for precision, as numbers are written at full precision either way, and setting it raises a warning to that effect.
`digits`	Deprecated and ignored. Decimals are written at whatever precision reads back as the same value, so there is no precision for this argument to control. It is ignored, and supplying it warns.

Value

NULL when writing to a file, otherwise a character string

Examples

```
ds_json <- dataset_json(
  iris,
  item_oid = "IG.IRIS",
  name = "IRIS",
  dataset_label = "Iris",
  columns = iris_items
)
nd <- write_dataset_ndjson(ds_json)

# Write to disk
write_dataset_ndjson(ds_json, tempfile(fileext = ".ndjson"))

# float_as_decimals writes floats as the "decimal" type, quoting the numbers.
# It is an interoperability choice for systems that require that type - it is
# not needed for precision, and setting it warns to say so.
nd <- suppressWarnings(write_dataset_ndjson(ds_json, float_as_decimals = TRUE))
```

Index

∗ **Dataset Metadata Setters**
    set_source_system, 11
∗ **datasets**
    iris_items, 6
    schema_1_1_0, 10
    schema_ndjson_1_1_0, 10

dataset_json, 2
datasetjson_example, 5

get_column_metadata, 5

iris_items, 6

read_dataset_dsjc, 7
read_dataset_json, 7
read_dataset_ndjson, 9

schema_1_1_0, 10
schema_ndjson_1_1_0, 10
set_dataset_label *(*set_source_system*)*, 11
set_dataset_name *(*set_source_system*)*, 11
set_file_oid *(*set_source_system*)*, 11
set_item_oid *(*set_source_system*)*, 11
set_last_modified *(*set_source_system*)*, 11
set_metadata_ref *(*set_source_system*)*, 11
set_metadata_version *(*set_source_system*)*, 11
set_originator *(*set_source_system*)*, 11
set_source_system, 11
set_study_oid *(*set_source_system*)*, 11
set_variable_attributes, 12

validate_dataset_dsjc, 13
validate_dataset_json, 14
validate_dataset_ndjson, 15

write_dataset_dsjc, 15
write_dataset_json, 17
write_dataset_ndjson, 18