

# Package: xportr (via r-universe)

November 3, 2024

**Title** Utilities to Output CDISC SDTM/ADaM XPT Files

**Version** 0.4.1

**Description** Tools to build CDISC compliant data sets and check for CDISC compliance.

**License** MIT + file LICENSE

**URL** <https://atorus-research.github.io/xportr/>,  
<https://github.com/atorus-research/xportr>

**BugReports** <https://github.com/atorus-research/xportr/issues>

**Depends** R (>= 3.5)

**Imports** checkmate, cli, dplyr (>= 1.0.2), glue (>= 1.4.2), haven (>= 2.5.0), lifecycle, magrittr, purrr (>= 0.3.4), readr, rlang (>= 0.4.10), stringr (>= 1.4.0), tidyselect

**Suggests** DT, knitr, labelled, metacore, pharmaverseadam, readxl, rmarkdown, testthat (>= 3.0.0), withr

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**Encoding** UTF-8

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.2

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

**RemoteUrl** <https://github.com/atorus-research/xportr>

**RemoteRef** HEAD

**RemoteSha** abdc49e680ff96132fe69ea6590cdc6abbb9b21c

## Contents

adsl_xportr . . . . .	2
dataset_spec . . . . .	4
var_spec . . . . .	4

xportr . . . . .	5
xportr_df_label . . . . .	7
xportr_format . . . . .	8
xportr_label . . . . .	10
xportr_length . . . . .	12
xportr_metadata . . . . .	13
xportr_options . . . . .	15
xportr_order . . . . .	16
xportr_split . . . . .	18
xportr_type . . . . .	19
xportr_write . . . . .	21
xpt_validate . . . . .	22

<b>Index</b>	<b>24</b>
--------------	-----------

---

adsl_xportr	<i>Analysis Dataset Subject Level</i>
-------------	---------------------------------------

---

## Description

An example dataset containing subject level data

## Usage

```
data("adsl_xportr")
```

## Format

adsl\_xportr:

A data frame with 306 rows and 51 columns:

**STUDYID** Study Identifier

**USUBJID** Unique Subject Identifier

**SUBJID** Subject Identifier for the Study

**RFSTDTC** Subject Reference Start Date/Time

**RFENDTC** Subject Reference End Date/Time

**RFXSTDTC** Date/Time of First Study Treatment

**RFXENDTC** Date/Time of Last Study Treatment

**RFICDTC** Date/Time of Informed Consent

**RFPENDTC** Date/Time of End of Participation

**DTHDTC** Date/Time of Death

**DTHFL** Subject Death Flag

**SITEID** Study Site Identifier

**AGE** Age

**AGEU** Age Units

**SEX** Sex

**RACE** Race  
**ETHNIC** Ethnicity  
**ARMCD** Planned Arm Code  
**ARM** Description of Planned Arm  
**ACTARMCD** Actual Arm Code  
**ACTARM** Description of Actual Arm  
**COUNTRY** Country  
**DMDTC** Date/Time of Collection  
**DMDY** Study Day of Collection  
**TRT01P** Planned Treatment for Period 01  
**TRT01A** Actual Treatment for Period 01  
**TRTSDTM** Datetime of First Exposure to Treatment  
**TRTSTMF** Time of First Exposure Imputation Flag  
**TRTEDTM** Datetime of Last Exposure to Treatment  
**TRTETMF** Time of Last Exposure Imputation Flag  
**TRTSDT** Date of First Exposure to Treatment  
**TRTEDT** Date of Last Exposure to Treatment  
**TRTDURD** Total Treatment Duration (Days)  
**SCRFDT** Screen Failure Date  
**EOSDT** End of Study Date  
**EOSSTT** End of Study Status  
**FRVDT** Final Retrieval Visit Date  
**RANDDT** Date of Randomization  
**DTHDT** Date of Death  
**DTHDTF** Date of Death Imputation Flag  
**DTHADY** Relative Day of Death  
**LDDTHELD** Elapsed Days from Last Dose to Death  
**LSTALVDT** Date Last Known Alive  
**SAFFL** Safety Population Flag  
**RACEGR1** Pooled Race Group 1  
**AGEGR1** Pooled Age Group 1  
**REGION1** Geographic Region 1  
**LDDTHGR1** Last Dose to Death - Days Elapsed Group 1  
**DTH30FL** Death Within 30 Days of Last Trt Flag  
**DTHA30FL** Death After 30 Days from Last Trt Flag  
**DTHB30FL** Death Within 30 Days of First Trt Flag

**Source**

Dataset created by `admiral::use_ad_template("adsl")`

---

dataset\_spec                      *Example Dataset Specification*

---

### Description

Example Dataset Specification

### Usage

```
data("dataset_spec")
```

### Format

dataset\_spec:

A data frame with 1 row and 9 columns:

**Dataset** chr: Dataset

**Description** chr: Dataset description

**Class** chr: Dataset class

**Structure** lgl: Logical, indicating if there's a specific structure

**Purpose** chr: Purpose of the dataset

**Key, Variables** chr: Join Key variables in the dataset

**Repeating** chr: Indicates if the dataset is repeating

**Reference Data** lgl: Reference Data

**Comment** chr: Additional comment

---

var\_spec                              *Example Dataset Variable Specification*

---

### Description

Example Dataset Variable Specification

### Usage

```
data("var_spec")
```

### Format

var\_spec:

A data frame with 216 rows and 19 columns:

**Order** Order of variable

**Dataset** Dataset

**Variable** Variable

**Label** Variable Label  
**Data Type** Data Type  
**Length** Variable Length  
**Significant Digits** Significant Digits  
**Format** Variable Format  
**Mandatory** Mandatory Variable Flag  
**Assigned Value** Variable Assigned Value  
**Codelist** Variable Codelist  
**Common** Common Variable Flag  
**Origin** Variable Origin  
**Pages** Pages  
**Method** Variable Method  
**Predecessor** Variable Predecessor  
**Role** Variable Role  
**Comment** Comment  
**Developer Notes** Developer Notes

---

 xportr

*Wrapper to apply all core xportr functions and write xpt*


---

## Description

Wrapper to apply all core xportr functions and write xpt

## Usage

```

xportr(
  .df,
  var_metadata = NULL,
  df_metadata = NULL,
  domain = NULL,
  verbose = NULL,
  path,
  strict_checks = FALSE
)
  
```

## Arguments

.df	A data frame of CDISC standard.
var_metadata	A data frame containing variable level metadata
df_metadata	A data frame containing dataset level metadata.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.

verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
path	Path where transport file will be written. File name sans will be used as xpt name.
strict_checks	If TRUE, xpt validation will report errors and not write out the dataset. If FALSE, xpt validation will report warnings and continue with writing out the dataset. Defaults to FALSE

### Value

Returns the input dataframe invisibly

### Examples

```
data("adsl_xportr", "dataset_spec", "var_spec")
adsl <- adsl_xportr

library(magrittr)
test_dir <- tempdir()

pipeline_path <- file.path(test_dir, "adslpipe.xpt")
xportr_path <- file.path(test_dir, "adslxptr.xpt")

dataset_spec_low <- setNames(dataset_spec, tolower(names(dataset_spec)))
names(dataset_spec_low)[[2]] <- "label"

var_spec_low <- setNames(var_spec, tolower(names(var_spec)))
names(var_spec_low)[[5]] <- "type"

adsl %>%
  xportr_metadata(var_spec_low, "ADSL", verbose = "none") %>%
  xportr_type() %>%
  xportr_length() %>%
  xportr_label() %>%
  xportr_order() %>%
  xportr_format() %>%
  xportr_df_label(dataset_spec_low) %>%
  xportr_write(pipeline_path)

# `xportr()` can be used to apply a whole pipeline at once
xportr(
  adsl,
  var_metadata = var_spec_low,
  df_metadata = dataset_spec_low,
  domain = "ADSL",
  verbose = "none",
  path = xportr_path
)
```

---

xportr_df_label	<i>Assign Dataset Label</i>
-----------------	-----------------------------

---

### Description

Assigns dataset label from a dataset level metadata to a given data frame. This is stored in the 'label' attribute of the dataframe.

### Usage

```
xportr_df_label(.df, metadata = NULL, domain = NULL, metacore = deprecated())
```

### Arguments

.df	A data frame of CDISC standard.
metadata	A data frame containing dataset. See 'Metadata' section for details.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
metacore	<b>[Deprecated]</b> Previously used to pass metadata now renamed with metadata

### Value

Data frame with label attributes.

### Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments two columns must be present:

1. Domain Name - passed as the 'xportr.df\_domain\_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Label Name - passed as the 'xportr.df\_label' option. Default: "label". Character values to update the 'label' attribute of the dataframe This is passed to haven::write\_xpt to note the label.

### Examples

```
adsl <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  SITEID = c(001, 002, 003),
  AGE = c(63, 35, 27),
  SEX = c("M", "F", "M")
)

metadata <- data.frame(
  dataset = c("adsl", "adae"),
```

```

  label = c("Subject-Level Analysis", "Adverse Events Analysis")
)

adsl <- xportr_df_label(adsl, metadata, domain = "adsl")

```

---

xportr\_format

*Assign SAS Format*


---

## Description

Assigns a SAS format from a variable level metadata to a given data frame. If no format is found for a given variable, it is set as an empty character vector. This is stored in the 'format.sas' attribute.

## Usage

```

xportr_format(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = NULL,
  metacore = deprecated()
)

```

## Arguments

.df	A data frame of CDISC standard.
metadata	A data frame containing variable level metadata. See 'Metadata' section for details.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
metacore	<b>[Deprecated]</b> Previously used to pass metadata now renamed with metadata

## Value

Data frame with SASformat attributes for each variable.

## Format Checks

This function carries out a series of basic checks to ensure the formats being applied make sense.

Note, the 'type' of message that is generated will depend on the value passed to the verbose argument: with 'stop' producing an error, 'warn' producing a warning, or 'message' producing a message. A value of 'none' will not output any messages.



1. If the variable has a suffix of DT, DTM, TM (indicating a numeric date/time variable) then a message will be shown if there is no format associated with it.
2. If a variable is character then a message will be shown if there is no \$ prefix in the associated format.
3. If a variable is character then a message will be shown if the associated format has greater than 31 characters (excluding the \$).
4. If a variable is numeric then a message will be shown if there is a \$ prefix in the associated format.
5. If a variable is numeric then a message will be shown if the associated format has greater than 32 characters.
6. All formats will be checked against a list of formats considered 'standard' as part of an ADaM dataset. Note, however, this list is not exhaustive (it would not be feasible to check all the functions within the scope of this package). If the format is not found in the 'standard' list, then a message is created advising the user to check.

Format Name	w Values	d Values
w.d	1 - 32	., 0 - 31
\$w.	1 - 200	
DATEw.	., 5 - 11	
DATETIMEw.	7 - 40	
DDMMYYw.	., 2 - 10	
HHMM.		
MMDDYYw.	., 2 - 10	
TIMEw.	., 2 - 20	
WEEKDATEw.	., 3 - 37	
YYMMDDw.	., 2 - 10	
B8601DAw.	., 8 - 10	
B8601DTw.d	., 15 - 26	., 0 - 6
B8601TM.		
IS8601DA.		
IS8601TM.		
E8601DAw.	., 10	
E8601DNw.	., 10	
E8601DTw.d	., 16 - 26	., 0 - 6
E8601DXw.	., 20 - 35	
E8601LXw.	., 20 - 35	
E8601LZw.	., 9 - 20	
E8601TMw.d	., 8 - 15	., 0 - 6
E8601TXw.	., 9 - 20	
E8601TZw.d	., 9 - 20	., 0 - 6

### Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain\_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Format Name - passed as the 'xportr.format\_name' option. Default: "format". Character values to update the 'format.sas' attribute of the column. This is passed to haven::write to note the format.
3. Variable Name - passed as the 'xportr.variable\_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.

## Examples

```
ads1 <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  BRTHDT = c(1, 1, 2)
)

metadata <- data.frame(
  dataset = c("ads1", "ads1"),
  variable = c("USUBJID", "BRTHDT"),
  format = c(NA, "DATE9.")
)

ads1 <- xportr_format(ads1, metadata, domain = "ads1")
```

---

xportr\_label

*Assign Variable Label*

---

## Description

Assigns variable label from a variable level metadata to a given data frame. This function will give detect if a label is greater than 40 characters which isn't allowed in XPT v5. If labels aren't present for the variable it will be assigned an empty character value. Labels are stored in the 'label' attribute of the column.

## Usage

```
xportr_label(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = NULL,
  metacore = deprecated()
)
```

## Arguments

.df	A data frame of CDISC standard.
metadata	A data frame containing variable level metadata. See 'Metadata' section for details.

domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
metacore	<b>[Deprecated]</b> Previously used to pass metadata now renamed with metadata

### Value

Data frame with label attributes for each variable.

### Messaging

label\_log() is the primary messaging tool for xportr\_label(). If there are any columns present in the '.df' that are not noted in the metadata, they cannot be assigned a label and a message will be generated noting the number of variables that have not been assigned a label.

If variables were not found in the metadata and the value passed to the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were missing in metadata.

### Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain\_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Variable Name - passed as the 'xportr.variable\_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
3. Variable Label - passed as the 'xportr.label' option. Default: "label". These character values to update the 'label' attribute of the column. This is passed to haven::write to note the label.

### Examples

```
adsl <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  SITEID = c(001, 002, 003),
  AGE = c(63, 35, 27),
  SEX = c("M", "F", "M")
)

metadata <- data.frame(
  dataset = "adsl",
  variable = c("USUBJID", "SITEID", "AGE", "SEX"),
  label = c("Unique Subject Identifier", "Study Site Identifier", "Age", "Sex")
)

adsl <- xportr_label(adsl, metadata, domain = "adsl")
```

---

xportr\_length

*Assign SAS Length*


---

### Description

Assigns the SAS length to a specified data frame, either from a metadata object or based on the calculated maximum data length. If a length isn't present for a variable the length value is set to maximum data length for character columns, and 8 for non-character columns. This value is stored in the 'width' attribute of the column.

### Usage

```
xportr_length(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = NULL,
  length_source = c("metadata", "data"),
  metacore = deprecated()
)
```

### Arguments

.df	A data frame of CDISC standard.
metadata	A data frame containing variable level metadata. See 'Metadata' section for details.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
length_source	Choose the assigned length from either metadata or data. If "metadata" is specified, the assigned length is from the metadata length. If "data" is specified, the assigned length is determined by the calculated maximum data length. <i>Permitted Values:</i> "metadata", "data"
metacore	<b>[Deprecated]</b> Previously used to pass metadata now renamed with metadata

### Value

Data frame with SAS default length attributes for each variable.

## Messaging

length\_log is the primary messaging tool for xportr\_length. If there are any columns present in the '.df' that are not noted in the metadata, they cannot be assigned a length and a message will be generated noting the number or variables that have not been assigned a length.

If variables were not found in the metadata and the value passed to the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were missing in the metadata.

## Metadata

The argument passed in the 'metadata' argument can either be a {metacore} object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain\_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Variable Name - passed as the 'xportr.variable\_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
3. Variable Label - passed as the 'xportr.length' option. Default: "length". These numeric values to update the 'width' attribute of the column. This is passed to haven::write to note the variable length.

## Examples

```
ads1 <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  BRTHDT = c(1, 1, 2)
)
```

```
metadata <- data.frame(
  dataset = c("ads1", "ads1"),
  variable = c("USUBJID", "BRTHDT"),
  length = c(10, 8)
)
```

```
ads1 <- xportr_length(ads1, metadata, domain = "ads1", length_source = "metadata")
```

---

xportr\_metadata      *Set variable specifications and domain*

---

## Description

Sets metadata and/or domain for a dataset in a way that can be accessed by other xportr functions. If used at the start of an xportr pipeline, it removes the need to set metadata and domain at each step individually. For details on the format of the metadata, see the 'Metadata' section for each function in question.

**Usage**

```
xportr_metadata(.df, metadata = NULL, domain = NULL, verbose = NULL)
```

**Arguments**

.df	A data frame of CDISC standard.
metadata	A data frame containing variable level metadata. See 'Metadata' section for details.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'

**Value**

.df dataset with metadata and domain attributes set

**Examples**

```
metadata <- data.frame(  
  dataset = "test",  
  variable = c("Subj", "Param", "Val", "NotUsed"),  
  type = c("numeric", "character", "numeric", "character"),  
  format = NA,  
  order = c(1, 3, 4, 2)  
)  
  
adlb <- data.frame(  
  Subj = as.character(123, 456, 789),  
  Different = c("a", "b", "c"),  
  Val = c("1", "2", "3"),  
  Param = c("param1", "param2", "param3")  
)  
  
xportr_metadata(adlb, metadata, "test")  
  
library(magrittr)  
  
adlb %>%  
  xportr_metadata(metadata, "test") %>%  
  xportr_type() %>%  
  xportr_order()
```

---

xportr_options	<i>Get or set xportr options</i>
----------------	----------------------------------

---

## Description

There are two mechanisms for working with options for xportr. One is the `options()` function, which is part of base R, and the other is the `xportr_options()` function, which is in the xportr package. The reason for these two mechanisms is has to do with legacy code and scoping.

The `options()` function sets options globally, for the duration of the R process. The `getOption()` function retrieves the value of an option. All xportr related options of this type are prefixed with "xportr.".

## Usage

```
xportr_options(...)
```

## Arguments

... Options to set, with the form name = value or a character vector of option names.

## Options with options()

**xportr.df\_domain\_name** defaults to "dataset": The name of the domain "name" column in dataset metadata.

**xportr.df\_label** defaults to "label": The column noting the dataset label in dataset metadata.

**xportr.domain\_name** defaults to "dataset": The name of the domain "name" column in variable metadata.

**xportr.variable\_name** defaults to "variable": The name of the variable "name" in variable metadata.

**xportr.type\_name** defaults to "type": The name of the variable type column in variable metadata.

**xportr.label** defaults to "label": The name of the variable label column in variable metadata.

**xportr.length** defaults to "length": The name of the variable length column in variable metadata.

**xportr.order\_name** defaults to "order": The name of the variable order column in variable metadata.

**xportr.format\_name** defaults to "format": The name of the variable format column in variable metadata.

**xportr.format\_verbose** defaults to "none": The default argument for the 'verbose' argument for `xportr_format`.

**xportr.label\_verbose** defaults to "none": The default argument for the 'verbose' argument for `xportr_label`.

**xportr.length\_verbose** defaults to "none": The default argument for the 'verbose' argument for `xportr_length`.

**xportr.type\_verbose** defaults to "label": The default argument for the 'verbose' argument for xportr\_type.

**xportr.character\_types** defaults to "character": The default character vector used to explicitly coerce R classes to character XPT types.

**xportr.character\_metadata\_types** defaults to c("character", "char", "text", "date", "posixct", "posixt", "datetime", "time", "partialdate", "partialtime", "partialdatetime", "incompletedatetime", "durationdatetime", "intervaldatetime"): The default character vector used to explicitly coerce R classes to character XPT types.

**xportr.numeric\_metadata\_types** defaults to c("integer", "numeric", "num", "float"): The default character vector used to explicitly coerce R classes to numeric XPT types.

**xportr.numeric\_types** defaults to c("integer", "float", "numeric", "posixct", "posixt", "time", "date"): The default character vector used to explicitly coerce R classes to numeric XPT types.

### Options with xportr\_options()

Alternative to the options(), the xportr\_options() function can be used to set the options. The xportr\_options() function also returns the current options when a character vector of the options keys are passed into it. If nothing is passed into it, it returns the state of all xportr options.

### Examples

```
xportr_options("xportr.df_label")
xportr_options(xportr.df_label = "data_label", xportr.label = "custom_label")
xportr_options(c("xportr.label", "xportr.df_label"))
xportr_options()
```

---

xportr\_order

*Order variables of a dataset according to Spec*

---

### Description

The dplyr::arrange() function is used to order the columns of the dataframe. Any variables that are missing an order value are appended to the end of the dataframe after all of the variables that have an order.

### Usage

```
xportr_order(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = NULL,
  metacore = deprecated()
)
```



**Arguments**

.df	A data frame of CDISC standard.
metadata	A data frame containing variable level metadata. See 'Metadata' section for details.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
verbose	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
metacore	<b>[Deprecated]</b> Previously used to pass metadata now renamed with metadata

**Value**

Dataframe that has been re-ordered according to spec

**Messaging**

var\_ord\_msg() is the primary messaging tool for xportr\_order(). There are two primary messages that are output from var\_ord\_msg(). The first is the "moved" variables. These are the variables that were not found in the metadata file and moved to the end of the dataset. A message will be generated noting the number, if any, of variables that were moved to the end of the dataset. If any variables were moved, and the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were moved.

The second primary message is the number of variables that were in the dataset, but not in the correct order. A message will be generated noting the number, if any, of variables that have been re-ordered. If any variables were reordered, and the 'verbose' argument is 'stop', 'warn', or 'message', a message will be generated detailing the variables that were reordered.

**Metadata**

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments three columns must be present:

1. Domain Name - passed as the 'xportr.domain\_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Variable Name - passed as the 'xportr.variable\_name' option. Default: "variable". This is used to match columns in '.df' argument and the metadata.
3. Variable Order - passed as the 'xportr.order\_name' option. Default: "order". These values used to arrange the order of the variables. If the values of order metadata are not numeric, they will be coerced to prevent alphabetical sorting of numeric values.

**Examples**

```
adsl <- data.frame(
  BRTHDT = c(1, 1, 2),
  STUDYID = c("mid987650", "mid987650", "mid987650"),
```

```
TRT01A = c("Active", "Active", "Placebo"),
USUBJID = c(1001, 1002, 1003)
)

metadata <- data.frame(
  dataset = c("adsl", "adsl", "adsl", "adsl"),
  variable = c("STUDYID", "USUBJID", "TRT01A", "BRTHDT"),
  order = 1:4
)

adsl <- xportr_order(adsl, metadata, domain = "adsl")
```

---

xportr\_split

*Deprecated - Split xpt file output*

---

## Description

### [Deprecated]

This function is *deprecated*. Please use the argument `max_gb_size` in the function `xportr_write()` instead.

Per the FDA Study Data Technical Conformance Guide(<https://www.fda.gov/media/88173/download>) section 3.3.2, dataset files sizes shouldn't exceed 5 GB. If datasets are large enough, they should be split based on a variable. For example, laboratory readings in ADLB can be split by LBCAT to split up hematology and chemistry data.

This function will tell `xportr_write()` to split the data frame based on the variable passed in `split_by`. When written, the file name will be prepended with a number for uniqueness. These files should be noted in the Reviewer Guides per CDISC guidance to note how you split your files.

## Usage

```
xportr_split(.df, split_by = NULL)
```

## Arguments

<code>.df</code>	A data frame of CDISC standard.
<code>split_by</code>	A quoted variable that will be passed to <code>base::split()</code> .

## Value

A data frame with an additional attribute added so `xportr_write()` knows how to split the data frame.

**Examples**

```
adlb <- data.frame(
  USUBJID = c(1001, 1002, 1003),
  LBCAT = c("HEMATOLOGY", "HEMATOLOGY", "CHEMISTRY")
)

adlb <- xportr_split(adlb, "LBCAT")
```

---

xportr\_type

*Coerce variable type*


---

**Description**

XPT v5 datasets only have data types of character and numeric. `xportr_type` attempts to collapse R classes to those two XPT types. The `'xportr.character_types'` option is used to explicitly collapse the class of a column to character using `as.character`. Similarly, `'xportr.numeric_types'` will collapse a column to a numeric type. If no type is passed for a variable, it is assumed to be numeric and coerced with `as.numeric()`.

**Usage**

```
xportr_type(
  .df,
  metadata = NULL,
  domain = NULL,
  verbose = NULL,
  metacore = deprecated()
)
```

**Arguments**

<code>.df</code>	A data frame of CDISC standard.
<code>metadata</code>	A data frame containing variable level metadata. See 'Metadata' section for details.
<code>domain</code>	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
<code>verbose</code>	The action this function takes when an action is taken on the dataset or function validation finds an issue. See 'Messaging' section for details. Options are 'stop', 'warn', 'message', and 'none'
<code>metacore</code>	<b>[Deprecated]</b> Previously used to pass metadata now renamed with <code>metadata</code>

**Details**

Certain care should be taken when using timing variables. R serializes dates based on a reference date of 01/01/1970 where XPT uses 01/01/1960. This can result in dates being 10 years off when outputting from R to XPT if you're using a date class. For this reason, `xportr` will try to determine what should happen with variables that appear to be used to denote time.

**Value**

Returns the modified table.

**Messaging**

`type_log()` is the primary messaging tool for `xportr_type()`. The number of column types that mismatch the reported type in the metadata, if any, is reported by `xportr_type()`. If there are any type mismatches, and the `'verbose'` argument is `'stop'`, `'warn'`, or `'message'`, each mismatch will be detailed with the actual type in the data and the type noted in the metadata.

**Metadata**

The argument passed in the `'metadata'` argument can either be a metacore object, or a `data.frame` containing the data listed below. If metacore is used, no changes to options are required.

For `data.frame` `'metadata'` arguments four columns must be present:

1. Domain Name - passed as the `'xportr.domain_name'` option. Default: "dataset". This is the column subset by the `'domain'` argument in the function.
2. Variable Name - passed as the `'xportr.variable_name'` option. Default: "variable". This is used to match columns in `'df'` argument and the metadata.
3. Variable Type - passed as the `'xportr.type_name'`. Default: "type". This is used to note the XPT variable "type" options are numeric or character.
4. (Option only) Character Types - The list of classes that should be explicitly coerced to a XPT Character type. Default: `c("character", "char", "text", "date", "posixct", "posixt", "datetime", "time", "partialdate", "partialtime", "partialdatetime", "incompletedatetime", "durationdatetime", "intervaldatetime")`
5. (Option only) Numeric Types - The list of classes that should be explicitly coerced to a XPT numeric type. Default: `c("integer", "numeric", "num", "float")`

**Examples**

```
metadata <- data.frame(
  dataset = "test",
  variable = c("Subj", "Param", "Val", "NotUsed"),
  type = c("numeric", "character", "numeric", "character")
)

.df <- data.frame(
  Subj = as.character(123, 456, 789),
  Different = c("a", "b", "c"),
  Val = c("1", "2", "3"),
  Param = c("param1", "param2", "param3")
)

df2 <- xportr_type(.df, metadata, "test")
```

---

xportr_write	<i>Write xpt v5 transport file</i>
--------------	------------------------------------

---

### Description

Writes a local data frame into SAS transport file of version 5. The SAS transport format is an open format, as is required for submission of the data to the FDA.

### Usage

```
xportr_write(
  .df,
  path,
  max_size_gb = NULL,
  metadata = NULL,
  domain = NULL,
  strict_checks = FALSE,
  label = deprecated()
)
```

### Arguments

.df	A data frame to write.
path	Path where transport file will be written. File name sans will be used as xpt name.
max_size_gb	Maximum size in GB of the exported file(s). If size of xpt file exceeds the specified maximum, it will split the data frame into multiple exported chunk(s).
metadata	A data frame containing dataset. See 'Metadata' section for details.
domain	Appropriate CDISC dataset name, e.g. ADAE, DM. Used to subset the metadata object.
strict_checks	If TRUE, xpt validation will report errors and not write out the dataset. If FALSE, xpt validation will report warnings and continue with writing out the dataset. Defaults to FALSE
label	<b>[Deprecated]</b> Previously used to to set the Dataset label. Use the metadata argument to set the dataset label.

### Details

- Variable and dataset labels are stored in the "label" attribute.
- SAS format are stored in the "SASformat" attribute.
- SAS type are based on the metadata attribute.

### Value

A data frame. `xportr_write()` returns the input data invisibly.

## Metadata

The argument passed in the 'metadata' argument can either be a metacore object, or a data.frame containing the data listed below. If metacore is used, no changes to options are required.

For data.frame 'metadata' arguments two columns must be present:

1. Domain Name - passed as the 'xportr.df\_domain\_name' option. Default: "dataset". This is the column subset by the 'domain' argument in the function.
2. Label Name - passed as the 'xportr.df\_label' option. Default: "label". Character values to update the 'label' attribute of the dataframe This is passed to haven::write\_xpt to note the label.

## Examples

```
adsl <- data.frame(
  SUBL = as.character(123, 456, 789),
  DIFF = c("a", "b", "c"),
  VAL = c("1", "2", "3"),
  PARAM = c("param1", "param2", "param3")
)

var_spec <- data.frame(
  dataset = "adsl",
  label = "Subject-Level Analysis Dataset",
  data_label = "ADSL"
)

xportr_write(adsl,
  path = paste0(tempdir(), "/adsl.xpt"),
  domain = "adsl",
  metadata = var_spec,
  strict_checks = FALSE
)
```

---

xpt\_validate

*Validate Dataset Can be Written to xpt*

---

## Description

Function used to validate dataframes before they are sent to haven::write\_xpt for writing.

## Usage

```
xpt_validate(data)
```

## Arguments

data                    Dataset to be exported as xpt file

*xpt\_validate*

23

**Value**

Returns a character vector of failed conditions

# Index

## \* datasets

- adsl\_xportr, 2
- dataset\_spec, 4
- var\_spec, 4

adsl\_xportr, 2

dataset\_spec, 4

getOption(), 15

options(), 15

var\_spec, 4

xportr, 5

xportr\_df\_label, 7

xportr\_format, 8

xportr\_label, 10

xportr\_length, 12

xportr\_metadata, 13

xportr\_options, 15

xportr\_order, 16

xportr\_split, 18

xportr\_type, 19

xportr\_write, 21

xpt\_validate, 22