Package ‘dataRetrieval’

February 27, 2025

Type Package

Title Retrieval Functions for USGS and EPA Hydrology and Water Quality Data

Version 2.7.18

Description Collection of functions to help retrieve U.S. Geological Survey and U.S. Environmental Protection Agency water quality and hydrology data from web services. Data are discovered from National Water Information System <https://waterservices.usgs.gov/> and <https://waterdata.usgs.gov/nwis>.
Water quality data are obtained from the Water Quality Portal <https://www.waterqualitydata.us/>.

License CC0

Copyright This software is in the public domain because it contains materials that originally came from the United States Geological Survey, an agency of the United States Department of Interior.

Depends R (>= 4.1.0)

Imports curl, lubridate (>= 1.5.0), stats, utils, xml2, readr (>= 1.4.0), jsonlite, httr2

Suggests covr, dplyr, knitr, rmarkdown, sf, testthat

Encoding UTF-8

BuildVignettes true

VignetteBuilder knitr

BugReports https://github.com/DOI-USGS/dataRetrieval/issues

RoxygenNote 7.3.2

NeedsCompilation no

Author Laura DeCicco [aut, cre] (<https://orcid.org/0000-0002-3915-9487>),
Robert Hirsch [aut] (<https://orcid.org/0000-0002-4534-075X>),
David Lorenz [aut],
Jordan Read [ctb],
Jordan Walker [ctb],

Lindsay Platt [ctb],
David Watkins [aut] (<https://orcid.org/0000-0002-7544-0700>),
David Blodgett [aut] (<https://orcid.org/0000-0001-9489-1710>),
Mike Johnson [aut] (<https://orcid.org/0000-0002-5288-8350>),
Aliesha Krall [ctb] (<https://orcid.org/0000-0003-2521-5043>),
Lee Stanish [ctb] (<https://orcid.org/0000-0002-9775-6861>)

Maintainer Laura DeCicco <ldecicco@usgs.gov>

Repository CRAN

Date/Publication 2025-02-27 14:50:02 UTC

Contents

`addWaterYear` *add a water year column*

Description

Add a column to the dataRetrieval data frame with the water year. WQP queries will return a water year column for the start and end dates of the data.

Usage

```
addWaterYear(rawData)
```

Arguments

`rawData`	the daily- or unit-values datset retrieved from NWISweb. Must have at least one of the following columns to add the new water year columns: 'dateTime', 'Date', 'ActivityStartDate', or 'ActivityEndDate'. The date column(s) can be character, POSIXct, Date. They cannot be numeric.

Value

data.frame with an additional integer column with "WY" appended to the date column name. For WQP, there will be 2 columns: 'ActivityStartDateWY' and 'ActivityEndDateWY'.

Examples

```
nwisData <- readNWISdv("04085427", "00060", "2022-01-01", "2022-06-30")
nwisData <- addWaterYear(nwisData)

wqpData <- readWQPqw("USGS-01594440", "01075", "", "")
wqpData <- addWaterYear(wqpData)
```

---

calcWaterYear *Extract WY from a date*

Description

Determine the correct water year based on a calendar date.

Usage

```
calcWaterYear(dateVec)
```

Arguments

`dateVec`	vector of dates as character ("YYYY-DD-MM"), Date, or POSIXct. Numeric does not work.

Details

This function calculates a water year based on the USGS definition that a water year starts on October 1 of the year before, and ends on September 30. For example, water year 2015 started on 2014-10-01 and ended on 2015-09-30.

Value

numeric vector indicating the water year

Examples

```
x <- seq(as.Date("2010-01-01"), as.Date("2010-12-31"), by = "month")
calcWaterYear(x)

y <- c("2010-01-01", "1994-02", "1980", "2009-11-01", NA)
calcWaterYear(y)
```

checkWQPdates *Date Check for Water Quality Portal*

Description

Checks date format for inputs to the Water Quality Portal. Used in `readWQPqw` and `readWQPdata`.

Usage

```
checkWQPdates(values)
```

Arguments

`values`	named list with arguments to send to the Water Quality Portal

Value

values named list with corrected arguments to send to the Water Quality Portal

Examples

```
values <- list(
  startDateLo = "01-01-2002",
  characteristicName = "Phosphorous",
  endDate = as.Date("2014-01-01")
)
values <- checkWQPdates(values)
```

check_param *Check values from codeservice*

Description

Call a service to check on values from: https://api.waterdata.usgs.gov/samples-data/codeservice/docs.

Usage

```
check_param(service = "characteristicgroup", ...)
```

Arguments

`service`	Options are: "characteristicgroup", "states", "counties", "countries", "sitetype", "samplemedia", "characteristics", "observedproperty"
`...`	Optional additional query arguments. Only "characteristics" and "observedproperty" have additional parameters options.

Value

List, structure depends on service.

Examples

```
groups <- check_param("characteristicgroup")
states <- check_param("states")
countries <- check_param("countries")
counties <- check_param("counties")
sitetypes <- check_param("sitetype")
samplemedia <- check_param("samplemedia")
characteristics <- check_param("characteristics",
                               group = "Biological")
observedProperties <- check_param("observedproperty",
                                  text = "phosphorus")
```

---

constructNWISURL *Construct NWIS url for data retrieval*

Description

Using USGS water web services to construct urls.

Usage

```
constructNWISURL(
  siteNumbers,
  parameterCd = "00060",
  startDate = "",
  endDate = "",
  service,
  statCd = "00003",
  format = "xml",
  expanded = TRUE,
  ratingType = "base",
  statReportType = "daily",
  statType = "mean"
)
```

Arguments

`siteNumbers`	string or vector of strings USGS site number. This is usually an 8 digit number
`parameterCd`	string or vector of USGS parameter code. This is usually an 5 digit number.
`startDate`	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record.
`endDate`	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record.
`service`	string USGS service to call. Possible values are "dv" (daily values), "uv" (unit/instantaneous values), "gwlevels" (groundwater),and "rating" (rating curve), "peak", "meas" (discrete streamflow measurements), "stat" (statistics web service BETA).
`statCd`	string or vector USGS statistic code only used for daily value service. This is usually 5 digits. Daily mean (00003) is the default.
`format`	string, can be "tsv" or "xml", and is only applicable for daily and unit value requests. "tsv" returns results faster, but there is a possibility that an incomplete file is returned without warning. XML is slower, but will offer a warning if the file was incomplete (for example, if there was a momentary problem with the internet connection). It is possible to safely use the "tsv" option, but the user must carefully check the results to see if the data returns matches what is expected. The default is therefore "xml".
`expanded`	logical defaults to `TRUE`. If `TRUE`, retrieves additional information, only applicable for qw data.
`ratingType`	can be "base", "corr", or "exsa". Only applies to rating curve data.
`statReportType`	character Only used for statistics service requests. Time division for statistics: daily, monthly, or annual. Default is daily. Note that daily provides statistics for each calendar day over the specified range of water years, i.e. no more than 366 data points will be returned for each site/parameter. Use `readNWISdata` or `readNWISdv` for daily averages. Also note that "annual" returns statistics for the calendar year. Use `readNWISdata` for water years. Monthly and yearly provide statistics for each month and year within the range individually.
`statType`	character Only used for statistics service requests. Type(s) of statistics to output for daily values. Default is mean, which is the only option for monthly and yearly report types. See the statistics service documentation at https://waterservices.usgs.gov/docs/statistics/ for a full list of codes.

Value

url string

Examples

```
site_id <- "01594440"
startDate <- "1985-01-01"
endDate <- ""
pCode <- c("00060", "00010")
url_daily <- constructNWISURL(site_id, pCode,
  startDate, endDate, "dv",
  statCd = c("00003", "00001")
)
url_unit <- constructNWISURL(site_id, pCode, "2012-06-28", "2012-06-30", "iv")

url_daily_tsv <- constructNWISURL(site_id, pCode, startDate, endDate, "dv",
  statCd = c("00003", "00001"), format = "tsv"
)
url_rating <- constructNWISURL(site_id, service = "rating", ratingType = "base")
url_peak <- constructNWISURL(site_id, service = "peak")
url_meas <- constructNWISURL(site_id, service = "meas")
url_gwl <- constructNWISURL(site_id, service = "gwlevels",
                            startDate = "2024-05-01", endDate = "2024-05-30")
```

constructUseURL *Construct URL for NWIS water use data service*

Description

Reconstructs URLs to retrieve data from here: https://waterdata.usgs.gov/nwis/wu

Usage

```
constructUseURL(years, stateCd, countyCd, categories)
```

Arguments

years	integer Years for data retrieval. Must be years ending in 0 or 5, or "ALL", which retrieves all available years.
stateCd	could be character (full name, abbreviation, id), or numeric (id)
countyCd	could be numeric (County IDs from countyCdLookup) or character ("ALL")
categories	character Two-letter cateogory abbreviation(s)

Value

url string

Examples

```
url <- constructUseURL(
  years = c(1990, 1995),
  stateCd = "Ohio",
  countyCd = c(1, 3),
  categories = "ALL"
)
```

constructWQPURL *Construct WQP url for data retrieval*

Description

Construct WQP url for data retrieval. This function gets the data from here: https://www.waterqualitydata.us

Usage

```
constructWQPURL(siteNumbers, parameterCd, startDate, endDate, legacy = TRUE)
```

Arguments

siteNumbers	string or vector of strings USGS site number.
parameterCd	string or vector of USGS parameter code. This is usually an 5 digit number.
startDate	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record.
endDate	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record.
legacy	Logical. If TRUE, uses legacy WQP services. Default is TRUE. Setting legacy = FALSE uses WQX3.0 WQP services, which are in-development, use with caution.

Value

url string

Examples

```
site_ids <- c("USGS-02292010", "USGS-02276877")
startDate <- "2020-01-01"
endDate <- ""
pCode <- c("80154", "00613")
url_wqp <- constructWQPURL(
  site_ids,
  pCode,
  startDate, endDate
)
url_wqp
charNames <- c(
  "Temperature",
  "Temperature, sample",
  "Temperature, water",
  "Temperature, water, deg F"
)
obs_url_orig <- constructWQPURL(
  siteNumbers = c(
    "IIDFG-41WSSPAHS",
    "USGS-02352560"
  ),
  parameterCd = charNames,
  startDate, ""
)
obs_url_orig
```

---

construct_USGS_sample_request

Construct request for USGS Samples Data

Description

This function creates the call for discrete water quality samples data service described at https://waterdata.usgs.gov/download-samples. Note: all possible arguments are included, but it is strongly recommended to only use the NECESSARY arguments. Leave unnecessary arguments as the default NA.

Usage

```
construct_USGS_sample_request(
  monitoringLocationIdentifier = NA,
  siteTypeCode = NA,
  boundingBox = NA,
  hydrologicUnit = NA,
  activityMediaName = NA,
  characteristicGroup = NA,
  characteristic = NA,
  characteristicUserSupplied = NA,
  activityStartDateLower = NA,
  activityStartDateUpper = NA,
  countryFips = NA,
  stateFips = NA,
  countyFips = NA,
  projectIdentifier = NA,
  recordIdentifierUserSupplied = NA,
  siteTypeName = NA,
  usgsPCode = NA,
  pointLocationLatitude = NA,
  pointLocationLongitude = NA,
  pointLocationWithinMiles = NA,
  dataType = "results",
  dataProfile = NA
)
```

Arguments

`monitoringLocationIdentifier`
: A monitoring location identifier has two parts: the agency code and the location number, separated by a dash (-). Location identifiers should be separated with commas, for example: AZ014-320821110580701, CAX01-15304600, USGS-040851385. Location numbers without an agency prefix are assumed to have the prefix USGS.

`siteTypeCode`
: Site type code query parameter. See available options by running `check_param("sitetype")$typeCode`.

`boundingBox`
: North and South are latitude values; East and West are longitude values. A vector of 4 (west, south, east, north) is expected. An example would be: c(-92.8, 44.2, -88.9, 46.0).

`hydrologicUnit`
: Hydrologic Unit Codes (HUCs) identify physical areas within the US that drain to a certain portion of the stream network. This filter accepts values containing 2, 4, 6, 8, 10 or 12 digits.

`activityMediaName`
: Sample media refers to the environmental medium that was sampled or analyzed.

`characteristicGroup`
: Characteristic group is a broad category describing the sample. See available options by running `check_param("characteristicgroup")$characteristicGroup`.

`characteristic`
: Characteristic is a specific category describing the sample. See available options by running `check_param("characteristics")$characteristicName`.

`characteristicUserSupplied`
: Observed property is the USGS term for the constituent sampled and the property name gives a detailed description of what was sampled. Observed property is mapped to characteristicUserSupplied and replaces the parameter name and pcode USGS previously used to describe discrete sample data. Find more information in the Observed Properties and Parameter Codes section of the Code Dictionary found here: `https://waterdata.usgs.gov/code-dictionary/`.

`activityStartDateLower`
: The service will return records with dates earlier than the value entered for activityStartDateUpper. Can be an R Date object, or a string with format YYYY-MM-DD. The logic is inclusive, i.e. it will also return records that match the date.

`activityStartDateUpper`
: The service will return records with dates later than the value entered for activityStartDateLower. Can be an R Date object, or a string with format YYYY-MM-DD. The logic is inclusive, i.e. it will also return records that match the date.

`countryFips`
: Country query parameter. Do not set redundant parameters. If another query parameter contains the country information, leave this parameter set to the default NA. See available options by running `check_param("countries")`, where the "id" field contains the value to use in the countryFips input.

`stateFips`
: State query parameter. To get a list of available state fips, run `check_param("states")`. The "fips" can be created using the function `stateCdLookup` - for example:

	`stateCdLookup("WI", "fips")`. FIPs codes for states take the format: CountryAbbrev:StateNumber, like US:55 for Wisconsin.
`countyFips`	County query parameter. To get a list of available counties, run `check_param("counties")`. The "Fips" can be created using the function `countyCdLookup` - for example: `countyCdLookup("WI", "Dane", "fips")` for Dane County, WI. FIPs codes for counties take the format: CountryAbbrev:StateNumber:CountyNumber, like US:55:025 for Dane County, WI.
`projectIdentifier`	Project identifier query parameter. This information would be needed from prior project information.
`recordIdentifierUserSupplied`	Record identifier, user supplied identifier. This information would be needed from the data supplier.
`siteTypeName`	Site type name query parameter. See available options by running `check_param("sitetype")$typeName`
`usgsPCode`	USGS parameter code. See available options by running `check_param("characteristics")$paramete`
`pointLocationLatitude`	Latitude for a point/radius query (decimal degrees). Must be used with pointLocationLongitude and pointLocationWithinMiles.
`pointLocationLongitude`	Longitude for a point/radius query (decimal degrees). Must be used with pointLocationLatitude and pointLocationWithinMiles.
`pointLocationWithinMiles`	Radius for a point/radius query. Must be used with pointLocationLatitude and pointLocationLongitude
`dataType`	Options include: "Results", "Monitoring locations", "Activities", "Projects", and "Organizations".
`dataProfile`	Profile depends on type. Options for "results" dataType are: "fullphyschem", "basicphyschem", "fullbio", "basicbio", "narrow", "resultdetectionquantitationlimit", "labsampleprep", "count". Options for "locations" are: "site" and "count". Options for "activities" are "sampact", "actmetric", "actgroup", and "count". Options for "projects" are: "project" and "projectmonitoringlocationweight". Options for "organizations" are: "organization" and "count".

Details

See also: `https://api.waterdata.usgs.gov/samples-data/docs`.

Value

data frame returned from web service call.

Examples

```
req <- construct_USGS_sample_request(
               monitoringLocationIdentifier = "USGS-04074950",
                characteristicUserSupplied = "pH, water, unfiltered, field")
rawData <- importWQP(req)
```

`countyCd` *US County Code Lookup Table*

Description

Data originally pulled from https://www2.census.gov/geo/docs/reference/codes/files/national_county.txt on April 1, 2015. On Feb. 11, 2022, the fields were updated with the file found in inst/extdata, which is used internally with NWIS retrievals.

Value

countyCd data frame.

Name	Type	Description
STUSAB	character	State abbreviation
STATE	character	two-digit ANSI code
COUNTY	character	three-digit county code
COUNTY_NAME	character	County full name
COUNTY_ID	character	County id

Examples

```
head(countyCd)
```

`countyCdLookup` *US county code look up*

Description

Function to simplify finding county and county code definitions. Used in `readNWISdata` and `readNWISuse`. Currently only has US counties.

Usage

```
countyCdLookup(state, county, outputType = "fips")
```

Arguments

`state`	could be character (full name, abbreviation, id), or numeric (id)
`county`	could be character (name, with or without "County") or numeric (id)
`outputType`	character can be "fullName", "tableIndex", "id", or "fullEntry".

Examples

```
fips <- countyCdLookup(state = "WI", county = "Dane")
id <- countyCdLookup(state = "WI", county = "Dane", outputType = "id")
name <- countyCdLookup(state = "OH", county = 13, output = "fullName")
entry <- countyCdLookup(state = "Pennsylvania", county = "ALLEGHENY COUNTY", output = "fullEntry")
fromIDs <- countyCdLookup(state = 13, county = 5, output = "fullName")
```

`create_NWIS_bib` *Create NWIS data citation*

Description

Uses attributes from the NWIS functions to create data citations.

Usage

```
create_NWIS_bib(x)
```

Arguments

`x`	Any data returned from an NWIS function, must include "queryTime" and "url" attributes, which should come with the data by default.

Details

See `?bibentry` for more information.

Value

bibentry object to use for citing the data.

Examples

```
nwisData <- readNWISdv("04085427", "00060", "2012-01-01", "2012-06-30")
nwis_citation <- create_NWIS_bib(nwisData)
nwis_citation

print(nwis_citation, style = "Bibtex")
print(nwis_citation, style = "citation")
```

`create_WQP_bib` *Create WQP data citation*

Description

Uses attributes from the WQP functions to create data citations.

Usage

```
create_WQP_bib(x)
```

Arguments

`x`	Any data returned from an NWIS function, must include "queryTime" and "url" attributes, which should come with the data by default.

Details

See `?bibentry` for more information.

Value

bibentry object to use for citing the data.

Examples

```
WQPData <- readWQPqw("USGS-05288705",
                     parameterCd = "00300")
wqp_citation <- create_WQP_bib(WQPData)
wqp_citation

print(wqp_citation, style = "Bibtex")
print(wqp_citation, style = "citation")
```

`findNLDI` *R Client for the Network Linked Data Index*

Description

Provides a formal client to the USGS Network Linked Data Index.

Usage

```
findNLDI(
  comid = NULL,
  nwis = NULL,
  wqp = NULL,
  huc12 = NULL,
  location = NULL,
  origin = NULL,
  nav = NULL,
  find = c("flowlines"),
  distance_km = 100,
  no_sf = FALSE,
  warn = TRUE
)
```

Arguments

`comid`	numeric or character. An NHDPlusV2 COMID
`nwis`	numeric or character. A USGS NWIS surface water siteID
`wqp`	numeric or character. A water quality point ID
`huc12`	numeric or character. A WBD HUC12 unit ID
`location`	numeric vector. Coordinate pair in WGS84 SRS ordered lng/lat (X,Y)
`origin`	named list. Specifying a feature type and ID (e.g. list("comid" = 101))
`nav`	character vector. where to navigate from the starting point. Options include along the upper mainsteam (UM), upstream tributary (UT), downstream mainstem (DM) and downstream divergences (DD). You may select one or more of the abbreviations ("UM", "UT", DM", "DD").
`find`	character vector. Define what resources to find along the navigation path(s) (see get_nldi_sources()$source). Can also include 'basin' or 'flowline', which will return the upstream basin of the starting feature or flowlines along the navigation respectively. The default is "flowlines". If you provide any other resource, AND want flowlines, then flowlines must be explicitly requested.
`distance_km`	numeric. Define how far to look along the navigation path in kilometers (default = 100)
`no_sf`	if available, should 'sf' be used for parsing, defaults to 'TRUE' if 'sf' is locally installed
`warn`	(default TRUE) should warnings be printed

Details

The function is useful for topology and location based feature discovery. A user must specify an origin feature, optional navigation direction(s) along the network, as well as features to identify along the navigated paths. Valid starting options can be given by one of the following arguments: comid, nwis, huc12, wqp, location, and start.

Value

a list of data.frames if sf is not installed, a list of sf objects if it is

Examples

```
# Find Features / Define origin features

## Find feature by COMID
findNLDI(comid = 101)

## Find feature by NWIS ID
findNLDI(nwis = "11120000")

## Find feature by WQP ID
findNLDI(wqp = "USGS-04024315")

## Find feature by LOCATION
findNLDI(location = c(-115, 40))

## GENERAL ORIGIN: COMID
findNLDI(origin = list("comid" = 101))

# Navigation (flowlines will be returned if find is unspecified)
# UPPER MAINSTEM of USGS-11120000
findNLDI(nwis = "11120000", nav = "UM")

# MULTI-REQUEST
# UPPER MAINSTEM and TRIBUTARY of USGS-11120000
findNLDI(nwis = "11120000", nav = c("UT", "UM"))

# Discover Features(flowlines will not be returned unless included in find)

## Find feature(s) on the upper tributary of USGS-11120000
findNLDI(nwis = "11120000", nav = "UT", find = c("nwis", "wqp"))

## Find upstream basin boundary and  of USGS-11120000
findNLDI(nwis = "11120000", find = "basin")

# Control Distance
## Limit search to 50 km
findNLDI(comid = 101, nav = "DM", find = c("nwis", "wqp", "flowlines"), distance_km = 50)
```

`getQuerySummary` *getting header information from a WQP query*

Description

getting header information from a WQP query

Usage

```
getQuerySummary(url)
```

Arguments

`url`	the query url

`getWebServiceData` *Function to return data from web services*

Description

This function accepts a url parameter, and returns the raw data.

Usage

```
getWebServiceData(obs_url, ...)
```

Arguments

`obs_url`	character containing the url for the retrieval
`...`	information to pass to header request

Details

To add a custom user agent, create an environmental variable: CUSTOM_DR_UA

Value

Returns xml, json, or text depending on the requested data.

Examples

```
siteNumber <- "02177000"
startDate <- "2012-09-01"
endDate <- "2012-10-01"
offering <- "00003"
property <- "00060"
obs_url <- constructNWISURL(siteNumber, property, startDate, endDate, "dv")

rawData <- getWebServiceData(obs_url)
```

`get_nldi_sources` *Get current NLDI offerings*

Description

Used to query the current resources available through the NLDI

Usage

```
get_nldi_sources(url = pkg.env$nldi_base)
```

Arguments

`url`	URL for NLDI sources. Default is supplied by package environment.

Value

data.frame

Examples

```
get_nldi_sources()
```

`importNGWMN` *Function to return data from the National Ground Water Monitoring Network waterML2 format*

Description

This function accepts a url parameter for a WaterML2 getObservation. This function is still under development, but the general functionality is correct.

Usage

```
importNGWMN(input, asDateTime = FALSE, tz = "UTC")
```

Arguments

`input`	character or raw, containing the url for the retrieval or a path to the data file, or raw XML.
`asDateTime`	logical, if `TRUE` returns date and time as POSIXct, if `FALSE`, character
`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided time zone offset. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

Value

mergedDF a data frame source, time, value, uom, uomTitle, comment, gmlID

Examples

```
params <- list(request = "GetObservation",
               service = "SOS",
               version = "2.0.0",
               observedProperty = "urn:ogc:def:property:OGC:GroundWaterLevel",
               responseFormat = "text/xml",
               featureOfInterest = "VW_GWDP_GEOSERVER.USGS.403836085374401")

obs_url <- httr2::request("https://cida.usgs.gov") |>
 httr2::req_url_path_append("ngwmn_cache") |>
 httr2::req_url_path_append("sos") |>
 httr2::req_url_query(!!!params)

#data_returned <- importNGWMN(obs_url)
```

---

importRDB1 *Function to return data from the NWIS RDB 1.0 format*

Description

This function accepts a url parameter that already contains the desired NWIS site, parameter code, statistic, startdate and enddate. It is not recommended to use the RDB format for importing multi-site data.

Usage

```
importRDB1(obs_url, asDateTime = TRUE, convertType = TRUE, tz = "UTC")
```

Arguments

`obs_url`	character containing the url for the retrieval or a file path to the data file.
`asDateTime`	logical, if `TRUE` returns date and time as POSIXct, if `FALSE`, Date
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character
`tz`	character to set timezone attribute of datetime. Default converts the datetimes to UTC (properly accounting for daylight savings times based on the data's provided tz_cd column). Recommended US values include "UTC", "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". For a complete list, see `https://en.wikipedia.org/wiki/List_of_tz_database_time_zones`

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
datetime	POSIXct	The date and time of the value converted to UTC (if asDateTime = `TRUE`
	character	or raw character string (if asDateTime = FALSE)
tz_cd	character	The time zone code for datetime
code	character	Any codes that qualify the corresponding value
value	numeric	The numeric value for the parameter
tz_cd_reported	The originally reported time zone	

Note that code and value are repeated for the parameters requested. The names are of the form XD_P_S, where X is literal, D is an option description of the parameter, P is the parameter code, and S is the statistic code (if applicable). If a date/time (dt) column contained incomplete date and times, a new column of dates and time was inserted. This could happen when older data was reported as dates, and newer data was reported as a date/time.

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned
comment	character	Header comments from the RDB file

Examples

```
site_id <- "02177000"
startDate <- "2012-09-01"
endDate <- "2012-10-01"
offering <- "00003"
property <- "00060"

obs_url <- constructNWISURL(site_id, property,
  startDate, endDate, "dv",
  format = "tsv"
)

data <- importRDB1(obs_url)


urlMultiPcodes <- constructNWISURL("04085427", c("00060", "00010"),
  startDate, endDate, "dv",
  statCd = c("00003", "00001"), "tsv"
)

multiData <- importRDB1(urlMultiPcodes)

unitDataURL <- constructNWISURL(site_id, property,
  "2020-10-30", "2020-11-01", "uv",
  format = "tsv"
) # includes timezone switch

unitData <- importRDB1(unitDataURL, asDateTime = TRUE)

iceSite <- "04024000"
start <- "2015-11-09"
end <- "2015-11-24"
urlIce <- constructNWISURL(iceSite, "00060", start, end, "uv", format = "tsv")

ice <- importRDB1(urlIce, asDateTime = TRUE)
iceNoConvert <- importRDB1(urlIce, convertType = FALSE)

# User file:
filePath <- system.file("extdata", package = "dataRetrieval")
fileName <- "RDB1Example.txt"
fullPath <- file.path(filePath, fileName)
importUserRDB <- importRDB1(fullPath)
```

`importWaterML1` *Function to return data from the NWISWeb WaterML1.1 service*

Description

This function accepts a url parameter that already contains the desired NWIS site, parameter code, statistic, startdate and enddate.

Usage

```
importWaterML1(obs_url, asDateTime = FALSE, tz = "UTC")
```

Arguments

`obs_url`	character or raw, containing the url for the retrieval or a file path to the data file, or raw XML.
`asDateTime`	logical, if `TRUE` returns date and time as POSIXct, if `FALSE`, Date
`tz`	character to set timezone attribute of datetime. Default converts the datetimes to UTC (properly accounting for daylight savings times based on the data's provided tz_cd column). Recommended US values include "UTC", "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". For a complete list, see `https://en.wikipedia.org/wiki/List_of_tz_database_time_zones`

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
	POSIXct	The date and time of the value converted to UTC (if asDateTime = TRUE),
	character	or raw character string (if asDateTime = FALSE)
tz_cd	character	The time zone code for
code	character	Any codes that qualify the corresponding value
value	numeric	The numeric value for the parameter

Note that code and value are repeated for the parameters requested. The names are of the form X_D_P_S, where X is literal, D is an option description of the parameter, P is the parameter code, and S is the statistic code (if applicable).

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
siteInfo	data.frame	A data frame containing information on the requested sites
variableInfo	data.frame	A data frame containing information on the requested parameters
statisticInfo	data.frame	A data frame containing information on the requested statistics on the data
queryTime	POSIXct	The time the data was returned

See Also

`renameNWISColumns`

Examples

```
site_id <- "02177000"
startDate <- "2012-09-01"
endDate <- "2012-10-01"
offering <- "00003"
property <- "00060"
obs_url <- constructNWISURL(site_id, property, startDate, endDate, "dv")

data <- importWaterML1(obs_url, asDateTime = TRUE)


unitDataURL <- constructNWISURL(
  site_id, property,
  "2013-11-03", "2013-11-03", "uv"
)
unitData <- importWaterML1(unitDataURL, TRUE)

# Two sites, two pcodes, one site has two data descriptors:
siteNumber <- c("01480015", "04085427")
obs_url <- constructNWISURL(
  siteNumber, c("00060", "00010"),
  startDate, endDate, "dv"
)
data <- importWaterML1(obs_url)
data$dateTime <- as.Date(data$dateTime)
data <- renameNWISColumns(data)
names(attributes(data))
attr(data, "url")
attr(data, "disclaimer")

inactiveSite <- "05212700"
inactiveSite <- constructNWISURL(inactiveSite, "00060",
                                 "2014-01-01", "2014-01-10", "dv")
inactiveSite <- importWaterML1(inactiveSite)

inactiveAndAcitive <- c("07334200", "05212700")
inactiveAndAcitive <- constructNWISURL(inactiveAndAcitive,
                         "00060", "2014-01-01", "2014-01-10", "dv")
inactiveAndAcitive <- importWaterML1(inactiveAndAcitive)

# Timezone change with specified local timezone:
tzURL <- constructNWISURL("04027000", c("00300", "63680"),
                          "2011-11-05", "2011-11-07", "uv")
tzIssue <- importWaterML1(tzURL,
  asDateTime = TRUE, tz = "America/Chicago"
)


filePath <- system.file("extdata", package = "dataRetrieval")
fileName <- "WaterML1Example.xml"
fullPath <- file.path(filePath, fileName)
importFile <- importWaterML1(fullPath, TRUE)
```

`importWaterML2` *Parse the WaterML2 timeseries portion of a waterML2 file*

Description

Returns data frame columns of all information with each time series measurement; Anything defined as a default, is returned as an attribute of that data frame.

Usage

```
importWaterML2(input, asDateTime = FALSE, tz = "UTC")
```

Arguments

`input`	XML with only the wml2:MeasurementTimeseries node and children
`asDateTime`	logical, if `TRUE` returns date and time as POSIXct, if `FALSE`, character
`tz`	character to set timezone attribute of datetime. Default is an empty quote, which converts the datetimes to UTC (properly accounting for daylight savings times based on the data's provided time zone offset). Possible values are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla"

Examples

```
baseURL <- httr2::request("https://waterservices.usgs.gov/nwis/dv")
baseURL <- httr2::req_url_query(baseURL,
                                format = "waterml,2.0",
                                sites = "01646500",
                                startDT = "2014-09-01",
                                endDT = "2014-09-08",
                                statCd = "00003",
                                parameterCd = "00060" )

timeseries <- importWaterML2(baseURL, asDateTime = TRUE, tz = "UTC")
```

importWQP *Basic Water Quality Portal Data parser*

Description

Imports data from the Water Quality Portal based on a specified url.

Usage

```
importWQP(obs_url, tz = "UTC", csv = TRUE, convertType = TRUE)
```

Arguments

obs_url	character URL to Water Quality Portal#' @keywords data import USGS web service
tz	character to set timezone attribute of datetime. Default is UTC (properly accounting for daylight savings times based on the data's provided tz_cd column). Possible values include "America/New_York","America/Chicago", "America/Denver","America/Los_An "America/Anchorage","America/Honolulu","America/Jamaica","America/Managua", "America/Phoenix", and "America/Metlakatla"
csv	logical. Is the data coming back with a csv or tsv format. Default is FALSE. Currently, the summary service does not support tsv, for other services tsv is the safer choice.
convertType	logical, defaults to TRUE. If TRUE, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character.

Value

retval dataframe raw data returned from the Water Quality Portal. Additionally, a POSIXct dateTime column is supplied for start and end times, and converted to UTC. See https://www.waterqualitydata.us/portal_userguide/ for more information.

See Also

readWQPdata, readWQPqw, whatWQPsites

Examples

```
# These examples require an internet connection to run

## Examples take longer than 5 seconds:

rawSampleURL <- constructWQPURL("USGS-01594440", "01075", "", "")

rawSample <- importWQP(rawSampleURL)
```

```
STORETex <- constructWQPURL("WIDNR_WQX-10032762", "Specific conductance", "", "")

STORETdata <- importWQP(STORETex)

STORETdata_char <- importWQP(STORETex, convertType = FALSE)
```

`is_dataRetrieval_user` *Is this a dataRetrieval user*

Description

Reveals if this is a user or not

Usage

```
is_dataRetrieval_user()
```

Examples

```
is_dataRetrieval_user()
```

`parameterCdFile` *List of USGS parameter codes*

Description

Complete list of USGS parameter codes as of Oct. 24, 2024.

Value

parameterData data frame with information about USGS parameters.

Name	Type	Description
parameter_cd	character	5-digit USGS parameter code
parameter_group_nm	character	USGS parameter group name
parameter_nm	character	USGS parameter name
casrn	character	Chemical Abstracts Service (CAS) Registry Number
srsname	character	Substance Registry Services Name
parameter_units	character	Parameter units

Examples

```
head(parameterCdFile[, 1:2])
```

`parse_WQP` *Convert WQP columns to correct types*

Description

Takes the character results and converts to numeric and dates.

Usage

```
parse_WQP(retval, tz = "UTC")
```

Arguments

`retval`	Data frame from WQP
`tz`	character to set timezone attribute of datetime. Default is UTC (properly accounting for daylight savings times based on the associated "TimeZone" column). Possible values include "America/New_York","America/Chicago", "America/Denver","America/Los_Angeles", "America/Anchorage","America/Honolulu","America/Jamaica","A "America/Phoenix", and "America/Metlakatla"

Value

data frame retval with converted columns

Examples

```
# These examples require an internet connection to run
rawSampleURL <- constructWQPURL("USGS-01594440", "01075", "", "")

## Examples take longer than 5 seconds:



rawSample <- importWQP(rawSampleURL, convertType = FALSE)
convertedSample <- parse_WQP(rawSample)
```

`pcode_to_name` *Parameter code to characteristic name*

Description

This function is useful to fine what characteristic name, result sample fraction, unit code, and other parameters are mapped with USGS parameter codes. This information is useful for converting workflows from a more traditional NWIS water quality retrieval to a Water Quality Portal retrieval.

Usage

```
pcode_to_name(parameterCd = "all")
```

Arguments

`parameterCd`	character that contains the code for a character vector of 5-digit parameter codes. Default is "all" which will return a complete list of parameter codes that have been mapped to a characteristic name.

Value

a data frame with columns "parm_cd", "description", "characteristicname", "measureunitcode", "resultsamplefraction", "resulttemperaturebasis", "resultstatisticalbasis", "resulttimebasis", "resultweightbasis", "resultparticlesizebasis", "last_rev_dt"

Examples

```
pcodes <- c("00070", "00075", "00430", "52642")


all <- pcode_to_name()
some <- pcode_to_name(pcodes)
```

`readNGWMNdata` *Import data from the National Groundwater Monitoring Network.*

Description

Only water level data and site locations and names are currently available through the web service.

Usage

```
readNGWMNdata(service, ..., asDateTime = TRUE, tz = "UTC")
```

Arguments

`service`	char Service for the request - "observation" and "featureOfInterest" are implemented.
`...`	Other parameters to supply, namely `siteNumbers` or `bbox`
`asDateTime`	logical if `TRUE`, will convert times to POSIXct format. Currently defaults to `FALSE` since time zone information is not included.
`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided time zone offset. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

Examples

```
# one site
site <- "USGS.430427089284901"
#oneSite <- readNGWMNdata(siteNumbers = site, service = "observation")

# multiple sites
sites <- c("USGS.272838082142201", "USGS.404159100494601", "USGS.401216080362703")
# Very slow:
# multiSiteData <- readNGWMNdata(siteNumbers = sites, service = "observation")
# attributes(multiSiteData)

# non-USGS site
# accepts colon or period between agency and ID
site <- "MBMG:702934"
# data <- readNGWMNdata(siteNumbers = site, service = "featureOfInterest")

# bounding box
# bboxSites <- readNGWMNdata(service = "featureOfInterest", bbox = c(30, -102, 31, 99))
# retrieve  sites.  Set asDateTime to false since one site has an invalid date
# Very slow:
# bboxData <- readNGWMNdata(service = "observation", siteNumbers = bboxSites$site[1:3],
#                            asDateTime = FALSE)
```

`readNGWMNlevels` *Retrieve groundwater levels from the National Ground Water Monitoring Network.*

Description

Retrieve groundwater levels from the National Ground Water Monitoring Network.

Usage

```
readNGWMNlevels(siteNumbers, asDateTime = TRUE, tz = "UTC")
```

Arguments

`siteNumbers`	character Vector of feature IDs formatted with agency code and site number separated by a period or semicolon, e.g. `USGS.404159100494601`.
`asDateTime`	logical Should dates and times be converted to date/time objects, or returned as character? Defaults to `TRUE`. Must be set to `FALSE` if a site contains non-standard dates.
`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided time zone offset. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

Examples

```
# one site
site <- "USGS.430427089284901"
# oneSite <- readNGWMNlevels(siteNumbers = site)

# multiple sites
sites <- c("USGS:272838082142201", "USGS:404159100494601", "USGS:401216080362703")
# multiSiteData <- readNGWMNlevels(sites)

# non-USGS site
site <- "MBMG.103306"
# data <- readNGWMNlevels(siteNumbers = site, asDateTime = FALSE)

# site with no data returns empty data frame
noDataSite <- "UTGS.401544112060301"
# noDataSite <- readNGWMNlevels(siteNumbers = noDataSite)
```

`readNGWMNsites` *Retrieve site data from the National Ground Water Monitoring Network.*

Description

Retrieve site data from the National Ground Water Monitoring Network.

Usage

```
readNGWMNsites(siteNumbers)
```

Arguments

`siteNumbers`	character Vector of feature IDs formatted with agency code and site number separated by a period or semicolon, e.g. `USGS.404159100494601`.

Value

A data frame the following columns: #'

Name	Type	Description
site	char	Site FID
description	char	Site description
dec_lat_va, dec_lon_va	numeric	Site latitude and longitude

Examples

```
# one site
site <- "USGS.430427089284901"
#oneSite <- readNGWMNsites(siteNumbers = site)

# non-USGS site
site <- "MBMG.103306"
#siteInfo <- readNGWMNsites(siteNumbers = site)
```

`readNWISdata` *General Data Import from NWIS*

Description

Returns data from the NWIS web service. Arguments to the function should be based on `https://waterservices.usgs.gov` service calls. See examples below for ideas of constructing queries.

Usage

```
readNWISdata(..., asDateTime = TRUE, convertType = TRUE, tz = "UTC")
```

Arguments

`...`	see `https://waterservices.usgs.gov/docs/site-service/` for a complete list of options. A list of arguments can also be supplied. One important argument to include is "service". Possible values are "iv" (for instantaneous), "dv" (for daily values), "gwlevels" (for groundwater levels), "site" (for site service), "measurement", and "stat" (for statistics service). Note: "measurement" calls go to: `https://nwis.waterdata.usgs.gov/usa/nwis` for data requests, and use different call requests schemes. The statistics service has a limited selection of arguments (see `https://waterservices.usgs.gov/docs/site-service/`).
`asDateTime`	logical, if `TRUE` returns date and time as POSIXct, if `FALSE`, Date
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character
`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided tz_cd column. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

Details

This function requires users to create their own arguments based on the NWIS web services. It is a more complicated function to use compared to other NWIS functions such as `readNWISdv`, `readNWISuv`, `readNWISgwl`, etc. However, this function adds a lot of flexibility to the possible queries. This function will also behave exactly as NWIS when it comes to date queries. NWIS by default will only return the latest value for the daily and instantaneous services. So if you do not provide a starting date, you will only get back the latest value. If you want the full period of record, you can use "startDate = '1900-01-01'". Other options for dates are periods, such as "period = 'P7D'" which translates to a period of 7 days. For period, use only a positive ISO-8601 duration format, which should not be expressed in periods of less than a day, or in increments of months M or years Y. period returns data for a site generally from now to a time in the past. Note that when period is used all data up to the most recent value are returned.

Value

A data frame with the following columns:

Name	Type	Description
agency	character	The NWIS code for the agency reporting the data
site	character	The USGS site number
dateTime	POSIXct	The date and time (if applicable) of the measurement, converted to UTC for unit value data. R only al
tz_cd	character	The time zone code for dateTime column
code	character	Any codes that qualify the corresponding value
value	numeric	The numeric value for the parameter

Note that code and value are repeated for the parameters requested. The names are of the form X_D_P_S, where X is literal, D is an option description of the parameter, P is the parameter code, and S is the statistic code (if applicable).

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
siteInfo	data.frame	A data frame containing information on the requested sites
variableInfo	data.frame	A data frame containing information on the requested parameters
statisticInfo	data.frame	A data frame containing information on the requested statistics on the data
queryTime	POSIXct	The time the data was returned

See Also

renameNWISColumns, importWaterML1, importRDB1

Examples

```
# Examples not run for time considerations

dataTemp <- readNWISdata(stateCd = "OH", parameterCd = "00010", service = "dv")
instFlow <- readNWISdata(
  sites = "05114000", service = "iv",
  parameterCd = "00060",
  startDate = "2014-05-01T00:00Z", endDate = "2014-05-01T12:00Z"
)

instFlowCDT <- readNWISdata(
  sites = "05114000", service = "iv",
  parameterCd = "00060",
  startDate = "2014-05-01T00:00", endDate = "2014-05-01T12:00",
  tz = "America/Chicago"
```

```
)

multiSite <- readNWISdata(
  sites = c("04025500", "040263491"),
  service = "iv", parameterCd = "00060"
)

bBoxEx <- readNWISdata(bBox = c(-83, 36.5, -81, 38.5), parameterCd = "00010")

startDate <- as.Date("2013-10-01")
endDate <- as.Date("2014-09-30")
waterYear <- readNWISdata(
  bBox = c(-83, 36.5, -82.5, 36.75),
  parameterCd = "00010",
  service = "dv",
  startDate = startDate,
  endDate = endDate
)

siteInfo <- readNWISdata(
  stateCd = "WI", parameterCd = "00010",
  hasDataTypeCd = "iv", service = "site"
)
temp <- readNWISdata(
  bBox = c(-83, 36.5, -82.5, 36.75), parameterCd = "00010", service = "site",
  seriesCatalogOutput = TRUE
)
GWL <- readNWISdata(site_no = c("392725077582401",
                                "375907091432201"),
                    parameterCd = "62610",
                    service = "gwlevels")

levels <- readNWISdata(stateCd = "WI",
                       service = "gwlevels",
                       startDate = "2024-05-01",
                       endDate = "2024-05-30")

meas <- readNWISdata(
  state_cd = "WI", service = "measurements",
  format = "rdb_expanded"
)

waterYearStat <- readNWISdata(
  site = c("01646500"),
  service = "stat",
  statReportType = "annual",
  statYearType = "water",
  missingData = "on"
)
monthlyStat <- readNWISdata(
  site = c("01646500"),
  service = "stat",
  statReportType = "monthly"
```

```
)

dailyStat <- readNWISdata(
  site = c("01646500"),
  service = "stat",
  statReportType = "daily",
  statType = c("p25", "p50", "p75", "min", "max"),
  parameterCd = "00060"
)

arg.list <- list(
  site = "03111548",
  statReportType = "daily",
  statType = c("p25", "p50", "p75", "min", "max"),
  parameterCd = "00060"
)
allDailyStats_2 <- readNWISdata(arg.list, service = "stat")

# use county names to get data
dailyStaffordVA <- readNWISdata(
  stateCd = "Virginia",
  countyCd = "Stafford",
  parameterCd = "00060",
  startDate = "2015-01-01",
  endDate = "2015-01-30"
)
va_counties <- c("51001", "51003", "51005", "51007", "51009", "51011", "51013", "51015")
va_counties_data <- readNWISdata(
  startDate = "2015-01-01", endDate = "2015-12-31",
  parameterCd = "00060", countycode = va_counties
)

site_id <- "01594440"
rating_curve <- readNWISdata(service = "rating", site_no = site_id, file_type = "base")
all_sites_base <- readNWISdata(service = "rating", file_type = "base")
all_sites_core <- readNWISdata(service = "rating", file_type = "corr")
all_sites_exsa <- readNWISdata(service = "rating", file_type = "exsa")
all_sites_24hrs <- readNWISdata(service = "rating", file_type = "exsa", period = 24)

peak_data <- readNWISdata(
  service = "peak",
  site_no = c("01594440", "040851325"),
  range_selection = "data_range"
)

peak_data <- readNWISdata(
  service = "peak",
  state_cd = "PA"
)

peak_data <- readNWISdata(
  service = "peak",
  huc2_cd = "20"
```

```
)
```

readNWISdv *Daily Value USGS NWIS Data Retrieval*

Description

Imports data from NWIS daily web service. This function gets the data from here: https://waterservices.usgs.gov/docs/dv-service/daily-values-service-details/ Inputs to this function are just USGS site ids, USGS parameter codes, USGS statistic codes, and start and end date. For a more complex query, use readNWISdata, with an argument service = "dv". Data coming the daily web services are aggregates of the instantaneous (sensor) web services. Not all statistical codes are available for all data. Use the function whatNWISdata to discover what data is available for a USGS site. The column data_type_cd with the values "dv" returned from whatNWISdata) are available from this service.

Usage

```
readNWISdv(
  siteNumbers,
  parameterCd,
  startDate = "",
  endDate = "",
  statCd = "00003"
)
```

Arguments

siteNumbers	character USGS site number. This is usually an 8 digit number. Multiple sites can be requested with a character vector.
parameterCd	character of USGS parameter code(s). This is usually an 5 digit number.
startDate	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record. Date arguments are always specified in local time.
endDate	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record. Date arguments are always specified in local time.
statCd	character USGS statistic code. This is usually 5 digits. Daily mean (00003) is the default.

Details

More information on the web service can be found here: https://waterservices.usgs.gov/test-tools, choosing the "Daily Value Service".

Value

A data frame with the following columns:

Name	Type	Description
agency	character	The NWIS code for the agency reporting the data
site	character	The USGS site number
Date	Date	The date of the value
code	character	Any codes that qualify the corresponding value
value	numeric	The numeric value for the parameter

Note that code and value are repeated for the parameters requested. The names are of the form X_D_P_S, where X is literal, D is an option description of the parameter, P is the parameter code, and S is the statistic code (if applicable).

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
siteInfo	data.frame	A data frame containing information on the requested sites
variableInfo	data.frame	A data frame containing information on the requested parameters
statisticInfo	data.frame	A data frame containing information on the requested statistics on the data
queryTime	POSIXct	The time the data was returned

See Also

renameNWISColumns, importWaterML1

Examples

```
site_id <- "04085427"
startDate <- "2012-01-01"
endDate <- "2012-06-30"
pCode <- "00060"

rawDailyQ <- readNWISdv(site_id, pCode, startDate, endDate)
rawDailyQAndTempMeanMax <- readNWISdv(site_id, c("00010", "00060"),
  startDate, endDate,
  statCd = c("00001", "00003")
)
rawDailyQAndTempMeanMax <- renameNWISColumns(rawDailyQAndTempMeanMax)
rawDailyMultiSites <- readNWISdv(c("01491000", "01645000"), c("00010", "00060"),
  startDate, endDate,
  statCd = c("00001", "00003")
)
# Site with no data:
x <- readNWISdv("10258500", "00060", "2014-09-08", "2014-09-14")
names(attributes(x))
attr(x, "siteInfo")
attr(x, "variableInfo")
```

```
site <- "05212700"
notActive <- readNWISdv(site, "00060", "2014-01-01", "2014-01-07")
```

`readNWISgwl` *Groundwater level measurements retrieval from USGS (NWIS)*

Description

Imports groundwater level data from NWIS web service. This function gets the data from here: https://waterservices.usgs.gov/docs/groundwater-levels/groundwater-levels-details/ Inputs to this function are just USGS site ids, USGS parameter codes, and start and end date. For a more complex query, use `readNWISdata`, including an argument service="gwlevels". Not all parameter codes are available for all data. Use the function `whatNWISdata` to discover what data is available for a USGS site. The column data_type_cd with the values "gw" returned from `whatNWISdata`) are available from this service.

Usage

```
readNWISgwl(
  siteNumbers,
  startDate = "",
  endDate = "",
  parameterCd = NA,
  convertType = TRUE,
  tz = "UTC"
)
```

Arguments

`siteNumbers`	character USGS site number (or multiple sites). This is usually an 8 digit number
`startDate`	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record.
`endDate`	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record.
`parameterCd`	character USGS parameter code. This is usually an 5 digit number. Default is "".
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character
`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided tz_cd column. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles",

"America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

Details

More information on the web service can be found here: `https://waterservices.usgs.gov/test-tools`, choosing the "Groundwater Levels Value Service".

Mixed date/times come back from the service depending on the year that the data was collected. See `https://waterdata.usgs.gov/usa/nwis/gw` for details about groundwater. By default the returned dates are converted to date objects, unless convertType is specified as FALSE. Sites with non-standard date formats (i.e. lacking a day) can be affected (see examples). See `https://waterservices.usgs.gov/docs/groundwater-levels/` for more information.

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
site_tp_cd	character	Site type code
lev_dt	Date	Date level measured
lev_tm	character	Time level measured
lev_tz_cd	character	Time datum
lev_va	numeric	Water level value in feet below land surface
sl_lev_va	numeric	Water level value in feet above specific vertical datum
lev_status_cd	character	The status of the site at the time the water level was measured
lev_agency_cd	character	The agency code of the person measuring the water level

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned
comment	character	Header comments from the RDB file
siteInfo	data.frame	A data frame containing information on the requested sites

See Also

`constructNWISURL`, `importRDB1`

Examples

```
site_id <- "434400121275801"
```

```
data <- readNWISgwl(site_id)
sites <- c("434400121275801", "375907091432201")
data2 <- readNWISgwl(sites, "", "")
data3 <- readNWISgwl("420125073193001", "", "")
# handling of data where date has no day
data4 <- readNWISgwl("425957088141001", startDate = "1980-01-01")

data5 <- readNWISgwl("263819081585801", parameterCd = "72019")
```

`readNWISmeas` *Surface-water measurement data retrieval from USGS (NWIS)*

Description

Reads surface-water measurement data from NWISweb. Data is retrieved from `https://waterdata.usgs.gov/nwis`. See `https://waterdata.usgs.gov/usa/nwis/sw` for details about surface water.

Usage

```
readNWISmeas(
  siteNumbers,
  startDate = "",
  endDate = "",
  tz = "UTC",
  expanded = FALSE,
  convertType = TRUE
)
```

Arguments

`siteNumbers`	character USGS site number (or multiple sites). This is usually an 8 digit number
`startDate`	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record.
`endDate`	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record.
`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided tz_cd column. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.
`expanded`	logical. Whether or not (TRUE or FALSE) to call the expanded data.

convertType	logical, defaults to TRUE. If TRUE, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character

Value

A data frame with at least the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
measurement_dt	POSIXct	The date and time (in POSIXct) of the measurement. Unless specified with the tz parameter, th
tz_cd	character	The time zone code for the measurement_dt column

See https://waterdata.usgs.gov/usa/nwis/sw for details about surface water, and https://waterdata.usgs.gov/nwis/help?output_formats_help for help on the columns and codes.

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned
comment	character	Header comments from the RDB file
siteInfo	data.frame	A data frame containing information on the requested sites
tz_cd_reported	The originally reported time zone	

See Also

constructNWISURL, importRDB1

Examples

```
site_ids <- c("01594440", "040851325")

data <- readNWISmeas(site_ids)
Meas05316840 <- readNWISmeas("05316840")
Meas05316840.ex <- readNWISmeas("05316840", expanded = TRUE)
Meas07227500.ex <- readNWISmeas("07227500", expanded = TRUE)
Meas07227500.exRaw <- readNWISmeas("07227500", expanded = TRUE, convertType = FALSE)
```

readNWISpCode — *USGS Parameter Data Retrieval*

Description

Imports data from NWIS about measured parameter based on user-supplied parameter code or codes. This function gets the data from here: https://nwis.waterdata.usgs.gov/nwis/pmcodes

Usage

```
readNWISpCode(parameterCd)
```

Arguments

`parameterCd`	character of USGS parameter codes (or multiple parameter codes). These are 5 digit number codes, more information can be found here: https://help.waterdata.usgs.gov/. To get a complete list of all current parameter codes in the USGS, use "all" as the input.

Value

parameterData data frame with the following information:

Name	Type	Description
parameter_cd	character	5-digit USGS parameter code
parameter_group_nm	character	USGS parameter group name
parameter_nm	character	USGS parameter name
casrn	character	Chemical Abstracts Service (CAS) Registry Number
srsname	character	Substance Registry Services Name
parameter_units	character	Parameter units

See Also

importRDB1

Examples

```
paramINFO <- readNWISpCode(c("01075", "00060", "00931"))
paramINFO <- readNWISpCode(c("01075", "00060", "00931", NA))

all_codes <- readNWISpCode("all")

one_extra <- readNWISpCode(c("01075", "12345"))
```

readNWISpeak *Peak flow data from USGS (NWIS)*

Description

Reads peak flow from NWISweb. Data is retrieved from https://waterdata.usgs.gov/nwis. In some cases, the specific date of the peak data is not know. This function will default to converting complete dates to a "Date" object, and converting incomplete dates to "NA". If those incomplete dates are needed, set the ‘asDateTime‘ argument to FALSE. No dates will be converted to R Date objects.

Usage

```
readNWISpeak(
  siteNumbers,
  startDate = "",
  endDate = "",
  asDateTime = TRUE,
  convertType = TRUE
)
```

Arguments

`siteNumbers`	character USGS site number(or multiple sites). This is usually an 8 digit number.
`startDate`	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record.
`endDate`	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record.
`asDateTime`	logical default to `TRUE`. When `TRUE`, the peak_dt column is converted to a Date object, and incomplete dates are removed. When `FALSE`, no columns are removed, but no dates are converted.
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
peak_dt	Date	Date of peak streamflow
peak_tm	character	Time of peak streamflow as character
peak_va	numeric	Annual peak streamflow value in cfs
peak_cd	character	Peak Discharge-Qualification codes (see `comment` for more information)

gage_ht	numeric	Gage height for the associated peak streamflow in feet
gage_ht_cd	character	Gage height qualification codes
year_last_pk	numeric	Peak streamflow reported is the highest since this year
ag_dt	Date	Date of maximum gage-height for water year (if not concurrent with peak)
ag_tm	character	Time of maximum gage-height for water year (if not concurrent with peak)
ag_gage_ht	numeric	maximum Gage height for water year in feet (if not concurrent with peak)
ag_gage_ht_cd	character	maximum Gage height code

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned
comment	character	Header comments from the RDB file
siteInfo	data.frame	A data frame containing information on the requested sites

See Also

constructNWISURL, importRDB1

Examples

```
site_ids <- c("01594440", "040851325")

data <- readNWISpeak(site_ids)
data2 <- readNWISpeak(site_ids, asDateTime = FALSE)
stations <- c("06011000")
peakdata <- readNWISpeak(stations, convertType = FALSE)
```

---

readNWISrating *Rating table for an active USGS streamgage retrieval*

Description

Reads current rating table for an active USGS streamgage from NWISweb. Data is retrieved from https://waterdata.usgs.gov/nwis.

Usage

```
readNWISrating(siteNumber, type = "base", convertType = TRUE)
```

Arguments

`siteNumber`	character USGS site number. This is usually an 8 digit number
`type`	character can be "base", "corr", or "exsa"
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character

Value

A data frame. If `type` is "base, " then the columns are INDEP, typically the gage height, in feet; DEP, typically the streamflow, in cubic feet per second; and STOR, where "*" indicates that the pair are a fixed point of the rating curve. If `type` is "exsa, " then an additional column, SHIFT, is included that indicates the current shift in the rating for that value of INDEP. If `type` is "corr, " then the columns are INDEP, typically the gage height, in feet; CORR, the correction for that value; and CORRINDEP, the corrected value for CORR.
If `type` is "base, " then the data frame has an attribute called "RATING" that describes the rating curve is included.

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned
comment	character	Header comments from the RDB file
siteInfo	data.frame	A data frame containing information on the requested sites
RATING	character	Rating information

Note

Not all active USGS streamgages have traditional rating curves that relate flow to stage.

See Also

`constructNWISURL`, `importRDB1`

Examples

```
site_id <- "01594440"

data <- readNWISrating(site_id, "base")
attr(data, "RATING")
```

`readNWISsite` *USGS Site File Data Retrieval*

Description

Imports data from USGS site file site. This function gets data from here: https://waterservices.usgs.gov/

Usage

```
readNWISsite(siteNumbers)
```

Arguments

`siteNumbers`	character USGS site number (or multiple sites). This is usually an 8 digit number

Value

A data frame with at least the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
station_nm	character	Site name
site_tp_cd	character	Site type
lat_va	numeric	DMS latitude
long_va	numeric	DMS longitude
dec_lat_va	numeric	Decimal latitude
dec_long_va	numeric	Decimal longitude
coord_meth_cd	character	Latitude-longitude method
coord_acy_cd	character	Latitude-longitude accuracy
coord_datum_cd	character	Latitude-longitude datum
dec_coord_datum_cd	character	Decimal Latitude-longitude datum
district_cd	character	District code
state_cd	character	State code
county_cd	character	County code
country_cd	character	Country code
land_net_ds	character	Land net location description
map_nm	character	Name of location map
map_scale_fc	character	Scale of location map
alt_va	numeric	Altitude of Gage/land surface
alt_meth_cd	character	Method altitude determined
alt_acy_va	numeric	Altitude accuracy
alt_datum_cd	character	Altitude datum
huc_cd	character	Hydrologic unit code
basin_cd	character	Drainage basin code
topo_cd	character	Topographic setting code

instruments_cd	character	Flags for instruments at site
construction_dt	character	Date of first construction
inventory_dt	character	Date site established or inventoried
drain_area_va	numeric	Drainage area
contrib_drain_area_va	numeric	Contributing drainage area
tz_cd	character	Time Zone abbreviation
local_time_fg	character	Site honors Daylight Savings Time
reliability_cd	character	Data reliability code
gw_file_cd	character	Data-other GW files
nat_aqfr_cd	character	National aquifer code
aqfr_cd	character	Local aquifer code
aqfr_type_cd	character	Local aquifer type code
well_depth_va	numeric	Well depth
hole_depth_va	numeric	Hole depth
depth_src_cd	character	Source of depth data
project_no	character	Project number

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned
comment	character	Header comments from the RDB file

Examples

```
siteINFO <- readNWISsite("05114000")
siteINFOMulti <- readNWISsite(c("05114000", "09423350"))
```

`readNWISstat` *Site statistics retrieval from USGS (NWIS)*

Description

Retrieves site statistics from the USGS Statistics Web Service beta. See `https://waterservices.usgs.gov/docs/statistics/` for more information.

Usage

```
readNWISstat(
  siteNumbers,
  parameterCd,
  startDate = "",
  endDate = "",
  convertType = TRUE,
  statReportType = "daily",
  statType = "mean"
)
```

Arguments

`siteNumbers`	character USGS site number (or multiple sites). This is usually an 8 digit number.
`parameterCd`	character USGS parameter code. This is usually a 5 digit number.
`startDate`	character starting date for data retrieval in the form YYYY, YYYY-MM, or YYYY-MM-DD. Dates cannot be more specific than the statReportType, i.e. startDate for monthly statReportTypes cannot include days, and annual statReportTypes cannot include days or months. Months and days are optional for the daily statReportType. Default is "" which indicates retrieval for the earliest possible record. For daily data, this indicates the start of the period the statistics will be computed over.
`endDate`	character ending date for data retrieval in the form YYYY, YYYY-MM, or YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record. For daily data, this indicates the end of the period the statistics will be computed over. The same restrictions as startDate apply.
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to numerics based on a standard algorithm. Years, months, and days (if appliccable) are also returned as numerics in separate columns. If convertType is false, everything is returned as a character.
`statReportType`	character time division for statistics: daily, monthly, or annual. Default is daily. Note that daily provides statistics for each calendar day over the specified range of water years, i.e. no more than 366 data points will be returned for each site/parameter. Use readNWISdata or readNWISdv for daily averages. Also note that 'annual' returns statistics for the calendar year. Use readNWISdata for water years. Monthly and yearly provide statistics for each month and year within the range indivually.
`statType`	character type(s) of statistics to output for daily values. Default is mean, which is the only option for monthly and yearly report types. See the statistics service documentation at `https://waterservices.usgs.gov/docs/statistics/` for a full list of codes.

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency repor
site_no	character	The USGS site number
parameter_cd	character	The USGS parameter code
Other columns will be present depending on statReportType and statType		

See Also

constructNWISURL, importRDB1

Examples

```
x1 <- readNWISstat(
  siteNumbers = c("02319394"),
  parameterCd = c("00060"),
  statReportType = "annual"
)

# all the annual mean discharge data for two sites
x2 <- readNWISstat(
  siteNumbers = c("02319394", "02171500"),
  parameterCd = c("00010", "00060"),
  statReportType = "annual"
)

# Request p25, p75, and mean values for temperature and discharge for the 2000s
# Note that p25 and p75 were not available for temperature, and return NAs
x <- readNWISstat(
  siteNumbers = c("02171500"),
  parameterCd = c("00010", "00060"),
  statReportType = "daily",
  statType = c("mean", "median"),
  startDate = "2000", endDate = "2010"
)
```

readNWISuse *Water use data retrieval from USGS (NWIS)*

Description

Retrieves water use data from USGS Water Use Data for the Nation. See https://waterdata.usgs.gov/nwis/wu for more information. All available use categories for the supplied arguments are retrieved.

Usage

```
readNWISuse(
  stateCd,
  countyCd,
  years = "ALL",
  categories = "ALL",
  convertType = TRUE,
  transform = FALSE
)
```

Arguments

`stateCd`	could be character (full name, abbreviation, id), or numeric (id). Only one is accepted per query.
`countyCd`	could be character (name, with or without "County", or "ALL"), numeric (id), or `NULL`, which will return state or national data depending on the stateCd argument. "ALL" may also be supplied, which will return data for every county in a state. Can be a vector of counties in the same state.
`years`	integer Years for data retrieval. Must be years ending in 0 or 5. Default is all available years.
`categories`	character categories of water use. Defaults to "ALL". Specific categories must be supplied as two- letter abbreviations as seen in the URL when using the NWIS water use web interface. Note that there are different codes for national and state level data.
`convertType`	logical defaults to `TRUE`. If `TRUE`, the function will convert the data to numerics based on a standard algorithm. Years, months, and days (if appliccable) are also returned as numerics in separate columns. If convertType is false, everything is returned as a character.
`transform`	logical only intended for use with national data. Defaults to `FALSE`, with data being returned as presented by the web service. If `TRUE`, data will be transformed and returned with column names, which will reformat national data to be similar to state data.

Value

A data frame with at least the year of record, and all available statistics for the given geographic parameters. County and state fields will be included as appropriate.

Examples

```
# All data for a county
allegheny <- readNWISuse(stateCd = "Pennsylvania", countyCd = "Allegheny")

# Data for an entire state for certain years
ohio <- readNWISuse(years = c(2000, 2005, 2010), stateCd = "OH", countyCd = NULL)

# Data for an entire state, county by county
```

```
pr <- readNWISuse(years = c(2000, 2005, 2010), stateCd = "PR", countyCd = "ALL")

# All national-scale data, transforming data frame to named columns from named rows
national <- readNWISuse(stateCd = NULL, countyCd = NULL, transform = TRUE)

# Washington, DC data
dc <- readNWISuse(stateCd = "DC", countyCd = NULL)

# data for multiple counties, with different input formatting
paData <- readNWISuse(stateCd = "42", countyCd = c("Allegheny County", "BUTLER", 1, "031"))

# retrieving two specific categories for an entire state
ks <- readNWISuse(stateCd = "KS", countyCd = NULL, categories = c("IT", "LI"))
```

readNWISuv *Instantaneous value data retrieval from USGS (NWIS)*

Description

Imports data from NWIS web service. This function gets the data from here: https://waterservices.usgs.gov/docs/instantaneous-values/instantaneous-values-details/ Inputs to this function are just USGS site ids, USGS parameter codes, and start and end date. For a more complex query, use readNWISdata, including an arguement service="uv". Not all parameter codes are available for all data. Use the function whatNWISdata to discover what data is available for a USGS site. The column data_type_cd with the values "uv" returned from whatNWISdata) are available from this service.

Usage

```
readNWISuv(siteNumbers, parameterCd, startDate = "", endDate = "", tz = "UTC")
```

Arguments

siteNumbers	character USGS site number (or multiple sites). This is usually an 8 digit number
parameterCd	character USGS parameter code. This is usually an 5 digit number.
startDate	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record. Simple date arguments are specified in local time. See more information here: https://waterservices.usgs.gov/docs/instantaneous-values/.
endDate	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record. Simple date arguments are specified in local time. See more information here: https://waterservices.usgs.gov/docs/instantaneous-values/.

`tz`	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided tz_cd column. Possible values to provide are "America/New_York", "America/Chicago", "America/Denver", "America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica", "America/Managua", "America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

Details

More information on the web service can be found here: `https://waterservices.usgs.gov/test-tools`, choosing the "Instantaneous Value Service".

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
dateTime	POSIXct	The date and time of the value converted to UTC
tz_cd	character	The time zone code for dateTime
code	character	Any codes that qualify the corresponding value
value	numeric	The numeric value for the parameter

Note that code and value are repeated for the parameters requested. The names are of the form: X_D_P_S, where X is literal, D is an option description of the parameter, P is the parameter code, and S is the statistic code (if applicable).

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
siteInfo	data.frame	A data frame containing information on the requested sites
variableInfo	data.frame	A data frame containing information on the requested parameters
statisticInfo	data.frame	A data frame containing information on the requested statistics on the data
queryTime	POSIXct	The time the data was returned

See Also

`renameNWISColumns`, `importWaterML1`

Examples

```
site_id <- "05114000"
parameterCd <- "00060"
startDate <- "2014-10-10"
```

```
endDate <- "2014-10-10"


rawData <- readNWISuv(site_id, parameterCd, startDate, endDate)

rawData_today <- readNWISuv(site_id, parameterCd, Sys.Date(), Sys.Date())

timeZoneChange <- readNWISuv(
  c("04024430", "04024000"), parameterCd,
  "2013-11-03", "2013-11-03"
)

centralTime <- readNWISuv(site_id, parameterCd,
  "2014-10-10T12:00", "2014-10-10T23:59",
  tz = "America/Chicago"
)

# Adding 'Z' to the time indicates to the web service to call the data with UTC time:
GMTdata <- readNWISuv(
  site_id, parameterCd,
  "2014-10-10T00:00Z", "2014-10-10T23:59Z"
)
```

readWQPdata — *General Data Import from Water Quality Portal*

Description

Imports data from Water Quality Portal web service. This function gets the data from here: https://www.waterqualitydata.us.

Usage

```
readWQPdata(
  ...,
  service = "Result",
  querySummary = FALSE,
  tz = "UTC",
  ignore_attributes = FALSE,
  convertType = TRUE
)
```

Arguments

`...`	see https://www.waterqualitydata.us/webservices_documentation for a complete list of options. A list of arguments can also be supplied. For more information see the above description for this help file. One way to figure out how

to construct a WQP query is to go to the "Advanced" form in the Water Quality Portal. Use the form to discover what parameters are available. Once the query is set in the form, scroll down to the "Query URL". You will see the parameters after "https://www.waterqualitydata.us/#". For example, if you chose "Nutrient" in the Characteristic Group dropdown, you will see characteristicType=Nutrient in the Query URL. The corresponding argument for dataRetrieval is characteristicType = "Nutrient". dataRetrieval users do not need to include mimeType, and providers is optional (these arguments are picked automatically).

`service` character. See Details for more information.

`querySummary` logical to only return the number of records and unique sites that will be returned from this query. Choosing TRUE is deprecated, readWQPsummary is recommended instead.

`tz` character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data's provided tz_cd column. Possible values to provide are "America/New_York","America/Chicago", "America/Denver","America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica","America/Managua","America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.

`ignore_attributes`
logical to choose to ignore fetching site and status attributes. Default is `FALSE`.

`convertType` logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character.

Details

This function uses ... as a query input, which can be very flexible, but also has a steeper learning curve. For a quick overview, scroll down to the Examples in this help file to see many query options.

There are currently 10 legacy options for data provided by the Water Quality Portal:

Legacy:

WQP Radio Button	service argument	Base URL
Sample Results	Result	/data/Result/search
Site Data Only	Station	/data/Station/search
Sampling Activity	Activity	/data/Activity/search
Sampling Activity Metrics	ActivityMetric	/data/ActivityMetric/search
Site Summary (not advertised on WQP)	SiteSummary	/data/summary/monitoringLocation/se
Project Data	Project	/data/Project/search
Project Monitoring Location Weighting Data	ProjectMonitoringLocationWeighting	/data/ProjectMonitoringLocationWeig
Result Detection Quantitation Limit Data	ResultDetectionQuantitationLimit	/data/ResultDetectionQuantitationLim
Biological Habitat Metrics	BiologicalMetric	/data/BiologicalMetric/search
Organization Data	Organization	/data/Organization/search

There are 4 WQX3 options. These are still in-development, and should be used with caution.

WQP Radio Button	service argument	Base URL	dataProfile
Monitoring Locations	StationWQX3	/wqx3/Station/search	
Full Physical Chemical	ResultWQX3	/wqx3/Result/search	fullPhysChem
Narrow	ResultWQX3	/wqx3/Result/search	narrow
Basic Physical Chemical	ResultWQX3	/wqx3/Result/search	basicPhysChem
Sampling Activity	ActivityWQX3	/wqx3/Activity/search	

Value

A data frame, the specific columns will depend on the "service" and/or "dataProfile".

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
siteInfo	data.frame	A data frame containing information on the requested sites
headerInfo	data.frame	A data frame returned from the WQP status service
queryTime	POSIXct	The time the data was returned

Examples

```
# Legacy:
nameToUse <- "pH"
pHData <- readWQPdata(siteid = "USGS-04024315",
                      characteristicName = nameToUse)
ncol(pHData)
attr(pHData, "siteInfo")
attr(pHData, "queryTime")
attr(pHData, "url")

# WQX3:
pHData_wqx3 <- readWQPdata(siteid = "USGS-04024315",
                      characteristicName = nameToUse,
                      service = "ResultWQX3",
                      dataProfile = "basicPhysChem")
attr(pHData_wqx3, "url")

# More examples:
# querying by county
DeWitt <- readWQPdata(
  statecode = "Illinois",
  countycode = "DeWitt",
  characteristicName = "Nitrogen"
)

attr(DeWitt, "url")

DeWitt_wqx3 <- readWQPdata(
  statecode = "Illinois",
  countycode = "DeWitt",
  characteristicName = "Nitrogen",
  service = "ResultWQX3",
  dataProfile = "basicPhysChem",
  ignore_attributes = TRUE)

attr(DeWitt_wqx3, "url")

# Data profile: "Sampling Activity"
activity <- readWQPdata(
  siteid = "USGS-04024315",
  service = "Activity"
)
attr(activity, "url")

activity_wqx3 <- readWQPdata(
  siteid = "USGS-04024315",
  service = "ActivityWQX3"
)
attr(activity_wqx3, "url")

Dane_activity <- readWQPdata(
  statecode = "Wisconsin",
  countycode = "Dane",
  startDateLo = "2023-01-01",
  startDateHi = "2023-12-31",
  service = "Activity"
)
attr(Dane_activity, "url")

Dane_activity_wqx3 <- readWQPdata(
  statecode = "Wisconsin",
  countycode = "Dane",
  startDateLo = "2023-01-01",
  startDateHi = "2023-12-31",
  service = "ActivityWQX3"
)
attr(Dane_activity_wqx3, "url")

########################################################
# Additional examples:


# Data profiles: "Organization Data"
org_data <- readWQPdata(
  statecode = "WI",
  countycode = "Dane",
  service = "Organization"
)

# Data profiles: "Project Data"
project_data <- readWQPdata(
  statecode = "WI",
  countycode = "Dane",
  service = "Project"
)

# Data profiles: "Project Monitoring Location Weighting Data"
proj_mlwd <- readWQPdata(
  statecode = "WI",
  countycode = "Dane",
  service = "ProjectMonitoringLocationWeighting"
)

# Data profiles: "Sample Results (physical/chemical metadata)"
samp_data <- readWQPdata(
  siteid = "USGS-04024315",
  dataProfile = "resultPhysChem",
  service = "Result"
)

# Data profiles: "Sample Results (biological metadata)"
samp_bio <- readWQPdata(
  siteid = "USGS-04024315",
  dataProfile = "biological",
  service = "Result"
)


# Data profiles: "Sample Results (narrow)"
samp_narrow <- readWQPdata(
  siteid = "USGS-04024315",
  service = "Result",
  dataProfile = "narrowResult"
)

samp_narrow_wqx3 <- readWQPdata(
  siteid = "USGS-04024315",
  service = "ResultWQX3",
  dataProfile = "narrow"
)


# Data profiles: "Sampling Activity"
samp_activity <- readWQPdata(
  siteid = "USGS-04024315",
  dataProfile = "activityAll",
  service = "Activity"
)

# Data profile: "Sampling Activity Metrics"
act_metrics <- readWQPdata(
  statecode = "WI",
  countycode = "Dane",
  service = "ActivityMetric"
```

```
)

# Data profile: "Result Detection Quantitation Limit Data"
dl_data <- readWQPdata(
  siteid = "USGS-04024315",
  service = "ResultDetectionQuantitationLimit"
)

# other options:
Phosphorus <- readWQPdata(
  statecode = "WI", countycode = "Dane",
  characteristicName = "Phosphorus",
  startDateLo = "2023-01-01",
  ignore_attributes = TRUE,
  convertType = FALSE
)

rawPHsites_legacy <- readWQPdata(siteid = c("USGS-05406450", "USGS-05427949", "WIDNR_WQX-133040"),
                        characteristicName = "pH",
                        service = "Result",
                        dataProfile = "narrowResult" )

rawPHsites <- readWQPdata(siteid = c("USGS-05406450", "USGS-05427949", "WIDNR_WQX-133040"),
                        characteristicName = "pH",
                        service = "ResultWQX3",
                        dataProfile = "narrow" )
```

readWQPqw — *Raw Data Import for Water Quality Portal*

Description

Imports data from the Water Quality Portal. This function gets the data from here: https://www.waterqualitydata.us. There are four required input arguments: siteNumbers, parameterCd, startDate, and endDate. parameterCd can either be a USGS 5-digit code, or a characteristic name. The sites can be either USGS, or other Water Quality Portal offered sites. It is required to use the 'full' site name, such as 'USGS-01234567'.

Usage

```
readWQPqw(
  siteNumbers,
  parameterCd,
  startDate = "",
  endDate = "",
  tz = "UTC",
  legacy = TRUE,
  querySummary = FALSE,
  ignore_attributes = FALSE,
  convertType = TRUE
)
```

Arguments

siteNumbers	character site number. This needs to include the full agency code prefix.
parameterCd	vector of USGS 5-digit parameter code or characteristicNames. Leaving this blank will return all of the measured values during the specified time period.
startDate	character starting date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the earliest possible record. Date arguments are always specified in local time.
endDate	character ending date for data retrieval in the form YYYY-MM-DD. Default is "" which indicates retrieval for the latest possible record. Date arguments are always specified in local time.
tz	character to set timezone attribute of dateTime. Default is "UTC", and converts the date times to UTC, properly accounting for daylight savings times based on the data provided tz_cd column. Possible values to provide are "America/New_York","America/Chicago", "America/Denver","America/Los_Angeles", "America/Anchorage", as well as the following which do not use daylight savings time: "America/Honolulu", "America/Jamaica","America/Managua","America/Phoenix", and "America/Metlakatla". See also `OlsonNames()` for more information on time zones.
legacy	Logical. If TRUE, uses legacy WQP services. Default is TRUE. Setting legacy = FALSE uses WQX3.0 WQP services, which are in-development, use with caution.
querySummary	logical to look at number of records and unique sites that will be returned from this query.
ignore_attributes	logical to choose to ignore fetching site and parameter attributes. Default is `FALSE`.
convertType	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character.

Value

A data frame derived from the default data profile.

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
siteInfo	data.frame	A data frame containing information on the requested sites
variableInfo	data.frame	A data frame containing information on the requested parameters

queryTime	POSIXct	The time the data was returned

See Also

readWQPdata, whatWQPsites, and importWQP

Examples

```
rawPcode <- readWQPqw("USGS-01594440", "01075", "", "")

attr(rawPcode, "siteInfo")
attr(rawPcode, "queryTime")
attr(rawPcode, "url")

rawCharacteristicName <- readWQPqw("WIDNR_WQX-10032762", "Specific conductance", "", "")
pHsites_legacy <- readWQPqw(c("USGS-05406450", "USGS-05427949", "WIDNR_WQX-133040"),
                        "pH", "", "")
ncol(pHsites_legacy)
attr(pHsites_legacy, "url")

pHsites_modern <- readWQPqw(c("USGS-05406450", "USGS-05427949", "WIDNR_WQX-133040"),
                        "pH", "", "", legacy = FALSE)
ncol(pHsites_modern)
attr(pHsites_modern, "url")

nwisEx <- readWQPqw("USGS-04024000", c("34247", "30234", "32104", "34220"), "", "2022-12-20")

DO <- readWQPqw(siteNumbers = "USGS-05288705",
                parameterCd = "00300",
                convertType = FALSE)
```

---

readWQPsummary   *Summary of Data Available from Water Quality Portal*

Description

Returns a list of sites with year-by-year information on what data is available. The function gets the data from: https://www.waterqualitydata.us. Arguments to the function should be based on https://www.waterqualitydata.us/webservices_documentation. The information returned from this function describes the available data at the WQP sites, and some metadata on the sites themselves.

Usage

```
readWQPsummary(...)
```

Arguments

`...`	see https://www.waterqualitydata.us/webservices_documentation for a complete list of options. A list of arguments can also be supplied. One way to figure out how to construct a WQP query is to go to the "Advanced" form in the Water Quality Portal: https://www.waterqualitydata.us/#mimeType=csv&providers=NWIS&providers=STORET Use the form to discover what parameters are available. Once the query is set in the form, scroll down to the "Query URL". You will see the parameters after "https://www.waterqualitydata.us/#". For example, if you chose "Nutrient" in the Characteristic Group dropdown, you will see characteristicType=Nutrient in the Query URL. The corresponding argument for dataRetrieval is characteristicType = "Nutrient". dataRetrieval users do not need to include mimeType, and providers is optional (these arguments are picked automatically).

Value

A data frame from the data returned from the Water Quality Portal about the data available for the query parameters.

See Also

whatWQPsites whatWQPdata

Examples

```
# Summary of a single site for the last 5 years:
site_5 <- readWQPsummary(
  siteid = "USGS-07144100",
  summaryYears = 5
)

# Summary of a single site for the full period of record:
site_all <- readWQPsummary(
  siteid = "USGS-07144100",
  summaryYears = "all"
)

# Summary of the data available from streams in a single county:
dane_county_data <- readWQPsummary(
  countycode = "US:55:025",
  summaryYears = 5,
  siteType = "Stream"
)

# Summary of the data all available from lakes in a single county:
lake_sites <- readWQPsummary(
  siteType = "Lake, Reservoir, Impoundment",
  countycode = "US:55:025"
)
```

```
# Summary of the data available for the last 5 years in New Jersey:
state1 <- readWQPsummary(
  statecode = "NJ",
  summaryYears = 5,
  siteType = "Stream"
)
```

`read_USGS_samples` *USGS Samples Data*

Description

This function creates the call and gets the data for discrete water quality samples data service described at `https://waterdata.usgs.gov/download-samples`.

Usage

```
read_USGS_samples(
  monitoringLocationIdentifier = NA,
  siteTypeCode = NA,
  boundingBox = NA,
  hydrologicUnit = NA,
  activityMediaName = NA,
  characteristicGroup = NA,
  characteristic = NA,
  characteristicUserSupplied = NA,
  activityStartDateLower = NA,
  activityStartDateUpper = NA,
  countryFips = NA,
  stateFips = NA,
  countyFips = NA,
  projectIdentifier = NA,
  recordIdentifierUserSupplied = NA,
  siteTypeName = NA,
  usgsPCode = NA,
  pointLocationLatitude = NA,
  pointLocationLongitude = NA,
  pointLocationWithinMiles = NA,
  dataType = "results",
  dataProfile = NA,
  tz = "UTC"
)
```

Arguments

`monitoringLocationIdentifier`	A monitoring location identifier has two parts: the agency code and the location number, separated by a dash (-). Location identifiers should be separated with commas, for example: AZ014-320821110580701, CAX01-15304600, USGS-040851385. Location numbers without an agency prefix are assumed to have the prefix USGS.
`siteTypeCode`	Site type code query parameter. See available options by running `check_param("sitetype")$typeCode`.
`boundingBox`	North and South are latitude values; East and West are longitude values. A vector of 4 (west, south, east, north) is expected. An example would be: c(-92.8, 44.2, -88.9, 46.0).
`hydrologicUnit`	Hydrologic Unit Codes (HUCs) identify physical areas within the US that drain to a certain portion of the stream network. This filter accepts values containing 2, 4, 6, 8, 10 or 12 digits.
`activityMediaName`	Sample media refers to the environmental medium that was sampled or analyzed.
`characteristicGroup`	Characteristic group is a broad category describing the sample. See available options by running `check_param("characteristicgroup")$characteristicGroup`.
`characteristic`	Characteristic is a specific category describing the sample. See available options by running `check_param("characteristics")$characteristicName`.
`characteristicUserSupplied`	Observed property is the USGS term for the constituent sampled and the property name gives a detailed description of what was sampled. Observed property is mapped to characteristicUserSupplied and replaces the parameter name and pcode USGS previously used to describe discrete sample data. Find more information in the Observed Properties and Parameter Codes section of the Code Dictionary found here: `https://waterdata.usgs.gov/code-dictionary/`.
`activityStartDateLower`	The service will return records with dates earlier than the value entered for activityStartDateUpper. Can be an R Date object, or a string with format YYYY-MM-DD. The logic is inclusive, i.e. it will also return records that match the date.
`activityStartDateUpper`	The service will return records with dates later than the value entered for activityStartDateLower. Can be an R Date object, or a string with format YYYY-MM-DD. The logic is inclusive, i.e. it will also return records that match the date.
`countryFips`	Country query parameter. Do not set redundant parameters. If another query parameter contains the country information, leave this parameter set to the default NA. See available options by running `check_param("countries")`, where the "id" field contains the value to use in the countryFips input.
`stateFips`	State query parameter. To get a list of available state fips, run `check_param("states")`. The "fips" can be created using the function `stateCdLookup` - for example:

	`stateCdLookup("WI", "fips")`. FIPs codes for states take the format: CountryAbbrev:StateNumber, like US:55 for Wisconsin.
`countyFips`	County query parameter. To get a list of available counties, run `check_param("counties")`. The "Fips" can be created using the function `countyCdLookup` - for example: `countyCdLookup("WI", "Dane", "fips")` for Dane County, WI. FIPs codes for counties take the format: CountryAbbrev:StateNumber:CountyNumber, like US:55:025 for Dane County, WI.
`projectIdentifier`	Project identifier query parameter. This information would be needed from prior project information.
`recordIdentifierUserSupplied`	Record identifier, user supplied identifier. This information would be needed from the data supplier.
`siteTypeName`	Site type name query parameter. See available options by running `check_param("sitetype")$typeName`
`usgsPCode`	USGS parameter code. See available options by running `check_param("characteristics")$paramete`
`pointLocationLatitude`	Latitude for a point/radius query (decimal degrees). Must be used with pointLocationLongitude and pointLocationWithinMiles.
`pointLocationLongitude`	Longitude for a point/radius query (decimal degrees). Must be used with pointLocationLatitude and pointLocationWithinMiles.
`pointLocationWithinMiles`	Radius for a point/radius query. Must be used with pointLocationLatitude and pointLocationLongitude
`dataType`	Options include: "Results", "Monitoring locations", "Activities", "Projects", and "Organizations".
`dataProfile`	Profile depends on type. Options for "results" dataType are: "fullphyschem", "basicphyschem", "fullbio", "basicbio", "narrow", "resultdetectionquantitationlimit", "labsampleprep", "count". Options for "locations" are: "site" and "count". Options for "activities" are "sampact", "actmetric", "actgroup", and "count". Options for "projects" are: "project" and "projectmonitoringlocationweight". Options for "organizations" are: "organization" and "count".
`tz`	character to set timezone attribute of datetime. Default is UTC (properly accounting for daylight savings times based on the data's provided tz_cd column). Possible values include "America/New_York","America/Chicago", "America/Denver","America/Los_An "America/Anchorage","America/Honolulu","America/Jamaica","America/Managua", "America/Phoenix", and "America/Metlakatla"

Examples

```
ph_data <- read_USGS_samples(
               monitoringLocationIdentifier = "USGS-04074950",
               characteristicUserSupplied = "pH, water, unfiltered, field",
               activityStartDateUpper = "2000-01-01",
                dataProfile = "narrow")

nameToUse <- "pH"
pHData <- read_USGS_samples(monitoringLocationIdentifier = "USGS-04024315",
                            characteristic = nameToUse)
ncol(pHData)
attr(pHData, "url")
attr(pHData, "queryTime")

summary_data <- read_USGS_samples(monitoringLocationIdentifier = "USGS-04024315",
                                  dataType = "projects")
```

renameNWISColumns *renameColumns*

Description

Rename columns coming back from NWIS data retrievals. Daily and unit value columns have names derived from their data descriptor, parameter, and statistic codes. This function reads information from the header and the arguments in the call to to rename those columns.

Usage

```
renameNWISColumns(
  rawData,
  p00010 = "Wtemp",
  p00045 = "Precip",
  p00060 = "Flow",
  p00065 = "GH",
  p00095 = "SpecCond",
  p00300 = "DO",
  p00400 = "pH",
  p62611 = "GWL",
  p63680 = "Turb",
  p72019 = "WLBLS",
  ...
)
```

Arguments

`rawData`	the daily- or unit-values datset retrieved from NWISweb.
`p00010`	the base name for parameter code 00010.
`p00045`	the base name for parameter code 00045.
`p00060`	the base name for parameter code 00060.

`p00065`	the base name for parameter code 00065.
`p00095`	the base name for parameter code 00095.
`p00300`	the base name for parameter code 00300.
`p00400`	the base name for parameter code 00400.
`p62611`	the base name for parameter code 62611.
`p63680`	the base name for parameter code 63680.
`p72019`	the base name for parameter code 72019.
`...`	named arguments for the base name for any other parameter code. The form of the name must be like pXXXXX, where XXXXX is the parameter code.

Value

A dataset like `data` with selected columns renamed.

Note

The following statistics codes are converted by `renameNWISColumns`.

00000 Instantaneous Value, suffix: Inst

00001 Maximum value, suffix: Max

00002 Minimum value, suffix: Min

00003 Mean value, no suffix

00006 Sum of values, suffix: Sum

00007 Modal value, suffix: Mode

00008 Median value, suffix: Median

00012 Equivalent mean value, suffix: EqMean

00021 Tidal high-high value, suffix: HiHiTide

00022 Tidal low-high value, suffix: LoHiTide

00023 Tidal high-low value, suffix: HiLoTide

00024 Tidal low-low value, suffix: LoLoTide

See Also

`readNWISdv`, `readNWISuv`

Examples

```
siteWithTwo <- "01480015"
startDate <- "2012-09-01"
endDate <- "2012-10-01"

twoResults <- readNWISdv(siteWithTwo, "00060", startDate, endDate)
names(twoResults)
renamedCols <- renameNWISColumns(twoResults)
names(renamedCols)
```

```
# Custom names:
newNames <- renameNWISColumns(twoResults, p00060 = "Discharge")
names(newNames)
```

setAccess *Set data endpoint*

Description

access Indicate which dataRetrieval access code you want to use options: `c('public','internal')`

Usage

```
setAccess(access = "public")
```

Arguments

`access`	code for data access. Options are: "public","internal","cooperator", or "USGS". • "internal" represents Access=3 ...for a single water science center • "USGS" represents Access=2 ...for all water science centers • "cooperator" represents Access=1 • "public" represents Access=0, public access

Author(s)

Luke Winslow, Jordan S Read

Examples

```
setAccess("internal")

setAccess("public")
```

`stateCd` *US State Code Lookup Table*

Description

Data originally pulled from `https://www2.census.gov/geo/docs/reference/state.txt` on April 1, 2015. On Feb. 11, 2022, the fields were updated with the file found in inst/extdata, which is used internally with NWIS retrievals.

Value

stateCd data frame.

Name	Type	Description
STATE	character	FIPS State Code
STUSAB	character	Official United States Postal Service (USPS) Code
STATE_NAME	character	State Name
STATENS	character	Geographic Names Information System Identifier (GNISID)

Examples

```
head(stateCd)
```

`stateCdLookup` *State code look up*

Description

Function to simplify finding state and state code definitions. Used in `readNWISdata` and `readWQPdata`.

Usage

```
stateCdLookup(input, outputType = "postal", country = "US")
```

Arguments

`input`	could be character (full name, abbreviation, id), or numeric (id)
`outputType`	character can be "postal", "fullName", "tableIndex", or "id".
`country`	description

Examples

```
fullName <- stateCdLookup("wi", "fullName")
abbriev <- stateCdLookup("Wisconsin", "postal")
id <- stateCdLookup("WI", "id")
name <- stateCdLookup(55, "fullName")
fips <- stateCdLookup("WI", "fips")
canada_st <- stateCdLookup(13, "fullName", country = "CA")
mexico_st <- stateCdLookup(13, "fullName", country = "MX")
stateCdLookup(c("West Virginia", "Wisconsin", 200, 55, "MN"))
```

`summarize_USGS_samples` *USGS Samples Summary Data*

Description

This function creates the call and gets the data for discrete water quality samples summary data service described at https://api.waterdata.usgs.gov/samples-data/docs.

Usage

```
summarize_USGS_samples(monitoringLocationIdentifier)
```

Arguments

`monitoringLocationIdentifier`	A monitoring location identifier has two parts, separated by a dash (-): the agency code and the location number. Location identifiers should be separated with commas, for example: AZ014-320821110580701, CAX01-15304600, USGS-040851385. Location numbers without an agency prefix are assumed to have the prefix USGS.

Value

data frame with summary of data available based on the monitoringLocationIdentifier

Examples

```
monitoringLocationIdentifier <- "USGS-04074950"

what_data <- summarize_USGS_samples(monitoringLocationIdentifier)
```

`whatNWISdata` *USGS data availability*

Description

Imports a table of available parameters, period of record, and count. See `https://waterservices.usgs.gov/docs/site-service/` for more information.

Usage

```
whatNWISdata(..., convertType = TRUE)
```

Arguments

`...`	see `https://waterservices.usgs.gov/docs/site-service/` for a complete list of options. A list of arguments can also be supplied.
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character

Details

This function requires users to create their own arguments based on the NWIS web services. It is a more complicated function to use compared to other NWIS functions such as `readNWISdv`, `readNWISuv`, etc. However, this function adds a lot of flexibility to the possible queries. If the "service" argument is included, the results will be filtered to the proper data_type_cd. This is a great function to use before a large data set, by filtering down the number of sites that have useful data.

Value

A data frame with the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
station_nm	character	Site name
site_tp_cd	character	Site type
dec_lat_va	numeric	Decimal latitude
dec_long_va	numeric	Decimal longitude
coord_acy_cd	character	Latitude-longitude accuracy
dec_coord_datum_cd	character	Decimal Latitude-longitude datum
alt_va	character	Altitude of Gage or land surface
alt_acy_va	character	Altitude accuracy
alt_datum_cd	character	Altitude datum
huc_cd	character	Hydrologic unit code
data_type_cd	character	Data type

parm_cd	character	Parameter code
stat_cd	character	Statistical code
dd_nu	character	Internal database key
loc_web_ds	character	Additional measurement description
medium_grp_cd	character	Medium group code
parm_grp_cd	character	Parameter group code
srs_id	character	SRS ID
access_cd	character	Access code
begin_date	Date	Begin date
end_date	Date	End date
count_nu	integer	Record count
parameter_group_nm	character	Parameter group name
parameter_nm	character	Parameter name
casrn	character	Chemical Abstracts Service (CAS) Registry Number
srsname	character	Substance Registry Services
parameter_units	character	Parameter units

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
comment	character	Header comments from the RDB file
queryTime	POSIXct	The time the data was returned

Examples

```
availableData <- whatNWISdata(siteNumber = "05114000")

# To find just unit value ('instantaneous') data:
uvData <- whatNWISdata(siteNumber = "05114000",
                       service = "uv")
uvDataMulti <- whatNWISdata(siteNumber = c("05114000", "09423350"),
                            service = c("uv", "dv"))
flowAndTemp <- whatNWISdata(
  stateCd = "WI", service = "dv",
  parameterCd = c("00060", "00010"),
  statCd = "00003"
)
sites <- whatNWISdata(stateCd = "WI",
                      parameterCd = "00060",
                      siteType = "ST",
                      service = "site")

sites <- whatNWISdata(stateCd = "WI",
                      service = "gwlevels")
```

`whatNWISsites` *Site Data Import from NWIS*

Description

Returns a list of sites from the NWIS web service. This function gets the data from: https://waterservices.usgs.gov/docs/site-service/. Mapper format is used

Usage

```
whatNWISsites(...)
```

Arguments

...	see https://waterservices.usgs.gov/docs/site-service/ for a complete list of options. A list (or lists) can also be supplied.

Value

A data frame with at least the following columns:

Name	Type	Description
agency_cd	character	The NWIS code for the agency reporting the data
site_no	character	The USGS site number
station_nm	character	Station name
site_tp_cd	character	Site type code
dec_lat_va	numeric	Decimal latitude
dec_long_va	numeric	Decimal longitude
queryTime	POSIXct	Query time

There are also several useful attributes attached to the data frame:

Name	Type	Description
url	character	The url used to generate the data
queryTime	POSIXct	The time the data was returned

Examples

```
siteListPhos <- whatNWISsites(stateCd = "OH", parameterCd = "00665")
oneSite <- whatNWISsites(sites = "05114000")
```

`whatWQPdata` *Data Available from Water Quality Portal*

Description

Returns a list of sites from the Water Quality Portal web service. This function gets the data from: `https://www.waterqualitydata.us`. Arguments to the function should be based on `https://www.waterqualitydata.us/webservices_documentation`. The information returned from whatWQPdata describes the available data at the WQP sites, and some metadata on the sites themselves. For example, a row is returned for each individual site that fulfills this query. In that we can learn how many sampling activities and results are available for the query. It does not break those results down by any finer grain. For example, if you ask for "Nutrients" (characteristicGroup), you will not learn what specific nutrients are available at that site. For that kind of data discovery see `readWQPsummary`.

Usage

```
whatWQPdata(..., convertType = TRUE)
```

Arguments

`...`	see `https://www.waterqualitydata.us/webservices_documentation` for a complete list of options. A list of arguments can also be supplied. One way to figure out how to construct a WQP query is to go to the "Advanced" form in the Water Quality Portal: `https://www.waterqualitydata.us/#mimeType=csv&providers=NWIS&providers=STORET` Use the form to discover what parameters are available. Once the query is set in the form, scroll down to the "Query URL". You will see the parameters after "https://www.waterqualitydata.us/#". For example, if you chose "Nutrient" in the Characteristic Group dropdown, you will see characteristicType=Nutrient in the Query URL. The corresponding argument for dataRetrieval is characteristicType = "Nutrient". dataRetrieval users do not need to include mimeType, and providers is optional (these arguments are picked automatically).
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character.

Value

A data frame that returns basic data availability such as sites, number of results, and number of sampling activities from the query parameters for the Water Quality Portal.

See Also

whatWQPsites readWQPsummary readWQPdata

Examples

```
site1 <- whatWQPdata(siteid = "USGS-01594440")

type <- "Stream"
sites <- whatWQPdata(countycode = "US:55:025", siteType = type)

lakeSites <- whatWQPdata(siteType = "Lake, Reservoir, Impoundment",
                         countycode = "US:55:025")
lakeSites_chars <- whatWQPdata(
  siteType = "Lake, Reservoir, Impoundment",
  countycode = "US:55:025", convertType = FALSE)


bbox <- c(-86.9736, 34.4883, -86.6135, 34.6562)
what_bb <- whatWQPdata(bBox = bbox)
```

`whatWQPsamples` *Site Data Import from Water Quality Portal*

Description

Returns a list of sites from the Water Quality Portal web service. This function gets the data from: `https://www.waterqualitydata.us`. Arguments to the function should be based on `https://www.waterqualitydata.us/webservices_documentation`. The return from this function returns the basic metadata on WQP sites. It is generally faster than the `whatWQPdata` function, but does not return information on what data was collected at the site.

Usage

```
whatWQPsamples(..., convertType = TRUE, legacy = TRUE)

whatWQPmetrics(..., convertType = TRUE)

whatWQPsites(..., legacy = TRUE, convertType = TRUE)
```

Arguments

`...`	see `https://www.waterqualitydata.us/webservices_documentation` for a complete list of options. A list of arguments can also be supplied. One way to figure out how to construct a WQP query is to go to the "Advanced" form in the Water Quality Portal: `https://www.waterqualitydata.us/#mimeType=csv&providers=NWIS&providers=STORET` Use the form to discover what parameters are available. Once the query is set in the form, scroll down to the "Query URL". You will see the parameters after "https://www.waterqualitydata.us/#". For example, if you chose "Nutrient" in the Characteristic Group dropdown, you

	will see characteristicType=Nutrient in the Query URL. The corresponding argument for dataRetrieval is characteristicType = "Nutrient". dataRetrieval users do not need to include mimeType, and providers is optional (these arguments are picked automatically).
`convertType`	logical, defaults to `TRUE`. If `TRUE`, the function will convert the data to dates, datetimes, numerics based on a standard algorithm. If false, everything is returned as a character.
`legacy`	Logical. If TRUE, uses legacy WQP services. Default is TRUE. Setting legacy = FALSE uses WQX3.0 WQP services, which are in-development, use with caution.

Value

A data frame with information on the sampling activity available from the Water Quality Portal for the query parameters.

data frame that includes information on site metadata.

See Also

whatWQPdata readWQPsummary

whatNWISdata

Examples

```
site1 <- whatWQPsamples(siteid = "USGS-01594440")

type <- "Stream"

sites <- whatWQPsamples(countycode = "US:55:025", siteType = type)

lakeSites_samples <- whatWQPsamples(siteType = "Lake, Reservoir, Impoundment",
                                    countycode = "US:55:025")



type <- "Stream"

sites <- whatWQPmetrics(countycode = "US:55:025", siteType = type)
lakeSites_metrics <- whatWQPmetrics(siteType = "Lake, Reservoir, Impoundment",
                                    countycode = "US:55:025")




site1 <- whatWQPsites(siteid = "USGS-01594440")

type <- "Stream"
sites <- whatWQPsites(
  countycode = "US:55:025",
  characteristicName = "Phosphorus",
  siteType = type
)
```

wqp_check_status *Get WQP service metadata*

Description

The information from this request is only available for a limited time after the original query from the WQP. In the readWQPdata and readWQPqw functions, the results from this function will be attached as an attribute to the data.

Usage

```
wqp_check_status(wqp_request_id)
```

Arguments

`wqp_request_id`	A character returned from the header of a WQP request.

Value

a list generated from the WQP describing what data was returned.

Examples

```
rawPcode <- readWQPqw("USGS-01594440", "01075",
                      ignore_attributes = TRUE, legacy = FALSE)
headerInfo <- attr(rawPcode, "headerInfo")
wqp_request_id <- headerInfo$`wqp-request-id`
count_info <- wqp_check_status(wqp_request_id)
count_info[["dataProviders"]]
```

zeroPad *Pad string with leading zeros*

Description

Function to pad a string with leading zeros. Useful for parameter codes and USGS site IDs.

Usage

```
zeroPad(x, padTo)
```

Arguments

`x`	character
`padTo`	number Final desired length of the character

Value

x character returned with leading zeros

Examples

```
pCode <- "10"
correctPCode <- zeroPad(pCode, 5)
pCodes <- c("100", "1000", "0", "12345", "1565465465465465")
correctPCodes <- zeroPad(pCodes, 5)
pCodeNA <- c(1, 2, NA)
padPCodeNA <- zeroPad(pCodeNA, 4)
```

Index