Package 'srppp'

July 17, 2025

Type Package

Title Read the Swiss Register of Plant Protection Products

Version 1.1.0

Date 2025-07-17

Description Generate data objects from XML versions of the Swiss Register of Plant Protection Products. An online version of the register can be accessed at <https://www.psm.admin.ch/de/produkte>. There is no guarantee of correspondence of the data read in using this package with that online version, or with the original registration documents. Also, the Federal Food Safety and Veterinary Office, coordinating the authorisation of plant protection products in Switzerland, does not answer requests regarding this package.

Depends R (>= 4.1.0), dm

Imports xml2, tibble, stringr, dplyr, tidyr, cli, data.tree, rlang

Suggests knitr, rmarkdown, here, DiagrammeR, DiagrammeRsvg, testthat (>= 3.0.0), parallel, waldo

BugReports https://github.com/agroscope-ch/srppp/issues

URL https://agroscope-ch.github.io/srppp/

License GPL (>= 3)

RoxygenNote 7.3.2.9000

VignetteBuilder knitr

Encoding UTF-8

Language en-GB

Config/testthat/edition 3

NeedsCompilation no

Author Daniel Baumgartner [ctb] (Provided feedback, explanations and background information),
Marcel Mathis [rev, ctb],
Romualdus Kasteel [rev] (Provided feedback to version 0.3.4),
Elisabeth Lutz [ctb],

Johannes Ranke [aut, cre] (ORCID:
<https://orcid.org/0000-0003-4371-6538>),
Agroscope [cph]

Maintainer Johannes Ranke <johannes.ranke@agroscope.admin.ch>

Repository CRAN

Date/Publication 2025-07-17 11:30:02 UTC

Contents

`alternative_products` *Find alternative products for all products containing certain active substances*

Description

This function searches for uses of a given list of active substances and reports either a table of uses with the number of available alternative products for each use, a detailed table of the alternative product uses, a table of uses without alternatives, or a list containing these three tables.

Usage

```
alternative_products(
  srppp,
  active_ingredients,
  details = FALSE,
  missing = FALSE,
  list = FALSE,
  lang = c("de", "fr", "it"),
  resolve_cultures = TRUE
)
```

Arguments

srppp	A srppp_dm object.
active_ingredients	Character vector of active ingredient names that will be matched against the column 'substances_de' in the srppp table 'substances'.
details	Should a table of alternative uses with 'wNbr' and 'use_nr' be returned?
missing	If this is set to TRUE, uses without alternative product registrations are listed.
list	If TRUE, a list of three tables is returned, a table of uses without alternative products ("Lückenindikationen"), a table of the number of alternative products for each use, if any, and a detailed table of all the alternative uses. This argument overrides the arguments 'details' and 'missing'.
lang	The language used for the active ingredient names and the returned tables.
resolve_cultures	Logical. Specifies whether to resolve culture levels to their most specific hierarchical level (leaf nodes) using a parent-child relationship dataset derived from a culture tree. • If TRUE (default), the function maps culture levels to their corresponding leaf nodes. This enables precise identification of alternative products at the most specific culture level. This resolves the problem that products are sometimes authorised for different cultural groups. This means that actual "Lückenindikationen" can be identified. Only supported in German, i.e. if lang = "de". • If FALSE, the function retains the original culture levels without hierarchical resolution. This option is useful when the original structure of the culture data needs to be preserved. **Note**: This argument is only applicable when the language is set to German (de). For other languages, the resolve_cultures functionality is not implemented and must be set to FALSE.

Details

A use is defined here as a combination of an application area, a crop ('culture') and a pathogen ('pest'). This means, that for an alternative product to be found, there has to be an exact match of application area, crop an pathogen.

Value

A tibble::tibble containing use definitions as defined above, i.e. containing columns with the application area, crop and pathogen. Depending on the arguments, columns summarizing or listing the alternative products and/or uses are also contained.

Examples

```
sr <- try(srppp_dm())

# Fall back to internal test data if downloading or reading fails
if (inherits(sr, "try-error")) {
```

```
  sr <- system.file("testdata/Daten_Pflanzenschutzmittelverzeichnis_2024-12-16.zip",
      package = "srppp") |>
    srppp_xml_get_from_path(from = "2024-12-16") |>
    srppp_dm()
}

# Examples with two active substances
actives_de <- c("Lambda-Cyhalothrin", "Deltamethrin")
alternative_products(sr, actives_de)
alternative_products(sr, actives_de, resolve_cultures = FALSE)
alternative_products(sr, actives_de, missing = TRUE)
alternative_products(sr, actives_de, details = TRUE)
alternative_products(sr, actives_de, list = TRUE)

# Examples resolving cultures
actives_de <- c("Spinetoram")
alternative_products(sr, actives_de, resolve_cultures = FALSE, list = TRUE)
alternative_products(sr, actives_de, resolve_cultures = TRUE, list = TRUE)

actives_de <- c("Schalenwicklergranulose-Virus")
alternative_products(sr, actives_de, resolve_cultures = FALSE, list = TRUE)
alternative_products(sr, actives_de, resolve_cultures = TRUE, list = TRUE)

actives_de <- c("Emamectinbenzoat")
alternative_products(sr, actives_de, resolve_cultures = FALSE, list = TRUE)
alternative_products(sr, actives_de, resolve_cultures = TRUE, list = TRUE)

# Example in Italian
actives_it <- c("Lambda-Cialotrina", "Deltametrina")
alternative_products(sr, actives_it, lang = "it", resolve_cultures = FALSE)
```

`application_rate_g_per_ha` *Calculate application rates for active ingredients*

Description

An application rate in g active substance/ha is calculated from information on dosage (product concentration in the application solution) and application volume, or directly from the product application rate. This is complicated by the fact that a rate ("expenditure" in the XML file) with units l/ha can refer to the application solution or to the liquid product.

Usage

```
application_rate_g_per_ha(
  product_uses,
  aggregation = c("max", "mean", "min"),
  dosage_units = c("percent_ww", "percent_vv", "state_of_matter"),
```

```
  skip_l_per_ha_without_g_per_L = TRUE,
  fix_l_per_ha = TRUE
)
```

Arguments

product_uses	A tibble containing the columns 'pNbr', 'use_nr', 'application_area_de', 'min_dosage', 'max_dosage', 'min_rate', 'max_rate', from the 'uses' table in a srppp_dm object, as well as the columns 'percent' and 'g_per_L' from the 'ingredients' table in a srppp_dm object.
aggregation	How to represent a range if present, e.g. "max" (default) or "mean".
dosage_units	If no units are given, or units are "%", then the applied amount in g/ha is calculated using a reference application volume and the dosage. As the dosage units are not explicitly given, we can specify our assumptions about these using this argument (currently not implemented, i.e. specifying the argument has no effect).
skip_l_per_ha_without_g_per_L	Per default, uses where the use rate has units of l/ha are skipped, if there is not product concentration in g/L. This was also done in the 2023 indicator project.
fix_l_per_ha	During the review of the 2023 indicator project calculations, a number of cases were identified where the unit l/ha specifies a water volume, and not a product volume. If TRUE (default), these cases are corrected, if FALSE, these cases are discarded.

Details

In some cases (currently one), external information was found, indicating that the "expenditure" is an application volume l_per_ha_is_water_volume.

Value

A tibble containing one additional column 'rate_g_per_ha'

Note

A reference application volume is used if there is no 'expenditure'. It is selected only based on the product application area. This is not correct if hops ('Hopfen') is the culture, as it has a unique reference application volume of 3000 L/ha.

Applications to hops were excluded for calculating mean use rates in the indicator project (Korkaric 2023), arguing that it is not grown in large areas in Switzerland.

Examples

```
library(srppp)
library(dplyr, warn.conflicts = FALSE)
library(dm, warn.conflicts = FALSE)

sr <- try(srppp_dm())
```

```
# Fall back to internal test data if downloading or reading fails
if (inherits(sr, "try-error")) {
  sr <- system.file("testdata/Daten_Pflanzenschutzmittelverzeichnis_2024-12-16.zip",
      package = "srppp") |>
    srppp_xml_get_from_path(from = "2024-12-16") |>
    srppp_dm()
}

product_uses_with_ingredients <- sr$substances |>
  filter(substance_de %in% c("Halauxifen-methyl", "Kupfer (als Kalkpr\u00E4parat)")) |>
  left_join(sr$ingredients, by = "pk") |>
  left_join(sr$uses, by = "pNbr") |>
  left_join(sr$products, by = "pNbr") |>
  select(pNbr, name, use_nr,
    min_dosage, max_dosage, min_rate, max_rate, units_de,
    application_area_de,
    substance_de, percent, g_per_L)

application_rate_g_per_ha(product_uses_with_ingredients) |>
  filter(name %in% c("Cerelex", "Pixxaro EC", "Bordeaux S")) |>
  select(ai = substance_de, app_area = application_area_de,
    min_d = min_dosage,  max_d = max_dosage,
    min_r = min_rate, max_r = max_rate,
    units_de, rate = rate_g_per_ha) |>
  print(n = Inf)
```

`l_per_ha_is_water_volume` *Use definitions where the rate in l/ha refers to the volume of the spraying solution*

Description

Use definitions where the rate in l/ha refers to the volume of the spraying solution

Usage

```
l_per_ha_is_water_volume
```

Format

An object of class `tbl_df` (inherits from `tbl`, `data.frame`) with 1 rows and 4 columns.

See Also

application_rate_g_per_ha

Examples

```
library(srppp)
l_per_ha_is_water_volume
```

`resolve_cultures` *Resolve culture specifications to their lowest hierarchical level*

Description

Resolves culture levels in a dataset to their lowest hierarchical level (leaf nodes) using a parent-child relationship dataset derived from a culture tree using the German culture names. Only German culture names are supported. If no match is found, the function assigns `NA` to the `leaf_culture_de` column. If `correct_culture_names` is set to `TRUE`, the function corrects variations in the naming of aggregated culture groups with "allg.".

Usage

```
resolve_cultures(
  dataset,
  srppp,
  culture_column = "culture_de",
  application_area_column = "application_area_de",
  correct_culture_names = TRUE,
  resolve_culture_allg = TRUE
)
```

Arguments

`dataset`	A data frame or tibble containing the data to be processed. It should include a column that represents the culture information to be resolved.
`srppp`	An srppp_dm object. From this object the relations from each culture to the leaf cultures (lowest level in the hierarchical tree) are used, which are stored as attribute 'culture_leaf_df' of the culture tree, which is itself stored as an attribute of the object.
`culture_column`	(Optional) A character string specifying the column in the dataset that contains the culture information to be resolved. Defaults to `"culture_de"`.
`application_area_column`	(Optional). A character string specifing the name of the column in dataset containing application area information. Default is "application_area_de".
`correct_culture_names`	If this argument is set to `TRUE`, the following corrections will be applied: In the `culture_tree`, and consequently in the `culture_leaf_df`, there are variations in the naming of aggregated culture groups with "allg.". For example, both "Obstbau allg." and "allg. Obstbau" exist. The information about the leaf nodes is only available in one of these terms. Therefore, the information from the term containing the leaf nodes is transferred to the corresponding "allg. ..." term.

`resolve_culture_allg`
: If this argument is set to `TRUE`, the culture "allg." is resolved to their lowest hierarchical level. The information on the application area is additionally used to resolve the culture "allg." to the lowest hierarchical level. For example if the culture "allg." is used in the application area "Obstbau", only the leaf cultures of the culture "Obstbau" are used to resolve the culture "allg.". If the application area is not found in the culture tree, all leaf cultures are used to resolve the culture "allg.".

Details

The `resolve_cultures` function processes the input dataset as follows

Leaf Node Resolution: The cultures in the specified column of the dataset are resolved to their lowest hierarchical level (leaf nodes) based on the `culture_leaf_df` mapping.

The result is an expanded dataset that includes an additional column (`leaf_culture_de`) containing the resolved cultures at their lowest level.

Value

A data frame or tibble with the same structure as the input `dataset`, but with an additional column `"leaf_culture_de"` that contains the resolved leaf culture levels. For cultures, that are not defined in the register, the leaf culture is set to `NA`.

Examples

```
library(srppp)
sr <- try(srppp_dm())

if (inherits(sr, "try-error")) {
  sr <- system.file("testdata/Daten_Pflanzenschutzmittelverzeichnis_2024-12-16.zip",
      package = "srppp") |>
    srppp_xml_get_from_path(from = "2024-12-16") |>
    srppp_dm()
}

example_dataset_1 <- data.frame(
  substance_de = c("Spirotetramat", "Spirotetramat", "Spirotetramat", "Spirotetramat"),
  pNbr = c(7839, 7839, 7839, 7839),
  use_nr = c(5, 7, 18, 22),
  application_area_de = c("Obstbau", "Obstbau", "Obstbau", "Obstbau"),
  culture_de = c("Birne", "Kirsche", "Steinobst", "Kernobst"),
   pest_de = c("Birnblattsauger", "Kirschenfliege", "Blattläuse (Röhrenläuse)", "Spinnmilben"))

# Same as above, but with culture name "Kirschen" instead of "Kirsche"
example_dataset_2 <- data.frame(
  substance_de = c("Spirotetramat", "Spirotetramat", "Spirotetramat", "Spirotetramat"),
  pNbr = c(7839, 7839, 7839, 7839),
  use_nr = c(5, 7, 18, 22),
  application_area_de = c("Obstbau", "Obstbau", "Obstbau", "Obstbau"),
  culture_de = c("Birne", "Kirschen", "Steinobst", "Kernobst"),
 pest_de = c("Birnblattsauger", "Kirschenfliege", "Blattläuse (Röhrenläuse)", "Spinnmilben"))
```

```
resolve_cultures(example_dataset_1, sr)

# Here we get NA for the leaf culture of "Kirschen"
resolve_cultures(example_dataset_2, sr)

# Example showing how cereals "Getreide" are resolved
example_dataset_3 <- data.frame(
  substance_de = c("Pirimicarb"),
  pNbr = c(2210),
  use_nr = c(3),
  application_area_de = c("Feldbau"),
  culture_de = c("Getreide"),
  pest_de = c("Blattläuse (Röhrenläuse)") )

resolve_cultures(example_dataset_3, sr)

# Example resolving ornamental plants ("Zierpflanzen")
example_dataset_4 <- data.frame(substance_de = c("Metaldehyd"),
 pNbr = 6142, use_nr = 1, application_area_de = c("Zierpflanzen"),
 culture_de = c("Zierpflanzen allg."), pest_de = c("Ackerschnecken/Deroceras Arten") )

resolve_cultures(example_dataset_4, sr)

# Illustrate the resolution of the culture "allg."
example_dataset_5 <- data.frame(
  substance_de = c("Kupfer (als Oxychlorid)","Metaldehyd","Metaldehyd","Schwefel"),
  pNbr = c(585,1090,1090,38),
  use_nr = c(12,4,4,1),
  application_area_de = c("Weinbau","Obstbau","Obstbau","Beerenbau"),
  culture_de = c("allg.","allg.","allg.","Brombeere"),
  pest_de = c("Graufäule (Botrytis cinerea)","Wegschnecken/Arion Arten",
    "Wegschnecken/Arion Arten","Gallmilben"))

 resolve_cultures(example_dataset_5, sr, resolve_culture_allg = FALSE)
 resolve_cultures(example_dataset_5, sr)

# Illustrate the resolution of "Obstbau allg.", which does not have children in
# the XML files, but which should have children, because Obstbau allg. is
# not a leaf culture.
example_dataset_6 <- data.frame(
  substance_de = c("Schwefel"),
  pNbr = c(3561),
  use_nr = c(4),
  application_area_de = c("Obstbau"),
  culture_de = c("Obstbau allg."),
  pest_de = c("Wühl- oder Schermaus") )

 resolve_cultures(example_dataset_6, sr,
   correct_culture_names = FALSE)
 resolve_cultures(example_dataset_6, sr,
   correct_culture_names = TRUE)
```

srppp_dm *Create a dm object from an XML version of the Swiss Register of Plant Protection Products*

Description

While reading in the data, the information obtained from the XML file is left unchanged, with the exceptions listed in the section 'Details'. An overview of the contents of the most important tables in the resulting data object is given in `vignette("srppp")`.

Usage

```
srppp_dm(from = srppp_xml_url, remove_duplicates = TRUE, verbose = TRUE)

## S3 method for class 'srppp_dm'
print(x, ...)
```

Arguments

`from`	A specification of the way to retrieve the XML to be passed to srppp_xml_get, or an object of the class 'srppp_xml'
`remove_duplicates`	Should duplicates based on wNbrs be removed?
`verbose`	Should we give some feedback?
`x`	A srppp_dm object
`...`	Not used

Details

Corrections made to the data:

- In the following case, the product composition is corrected while reading in the data: The active substance content of Dormex (W-3066) is not 667 g/L, but 520 g/L This was confirmed by a visit to the Wädenswil archive by Johannes Ranke and Daniel Baumgartner, 2024-03-27.

Removal of redundant information:

- Information on products that has been duplicated across several products sharing the same P-Number has been associated directly with this P-Number, in order to avoid duplications. While reading in the XML file, it is checked that the resulting deduplication does not remove any data.
- In very few cases of historical XML files, there are two `<Product>` sections sharing the same W-Number. In these cases, one of these has apparently been included in error and an informed decision is taken while reading in the data which one of these sections is discarded. The details of this procedure can be found in the source code of the function `srppp_xml_get_products`.

Amendments to the data:

In the table of obligations, the following information on mitigation measures is extracted from the ones relevant for the environment (SPe 3).

- "sw_drift_dist": Unsprayed buffer towards surface waters to mitigate spray drift in meters
- "sw_runoff_dist": Vegetated buffer towards surface waters to mitigate runoff in meters
- "sw_runoff_points": Required runoff mitigation points to mitigate runoff
- "biotope_drift_dist": Unsprayed buffer towards biotopes (as defined in articles 18a and 18b of the Federal Act on the Protection of Nature and Cultural Heritage) to mitigate spray drift in meters

Value

A dm::dm object with tables linked by foreign keys pointing to primary keys, i.e. with referential integrity. Since version 1.1, the returned object has an attribute named 'culture_tree' of class data.tree::Node.

Examples

```
 # Avoid NOTE on CRAN caused by checks >5s
library(dplyr, warn.conflicts = FALSE)
library(dm, warn.conflicts = FALSE)

sr <- try(srppp_dm())

# Fall back to internal test data if downloading or reading fails
if (inherits(sr, "try-error")) {
  sr <- system.file("testdata/Daten_Pflanzenschutzmittelverzeichnis_2024-12-16.zip",
      package = "srppp") |>
    srppp_xml_get_from_path(from = "2024-12-16") |>
    srppp_dm()
}

dm_examine_constraints(sr)
dm_draw(sr)

# Show ingredients for products named 'Boxer'
sr$products |>
  filter(name == "Boxer") |>
  left_join(sr$ingredients, by = "pNbr") |>
  left_join(sr$substances, by = "pk") |>
  select(wNbr, name, pNbr, isSalePermission, substance_de, g_per_L)

# Show authorised uses of the original product
boxer_uses <- sr$products |>
  filter(name == "Boxer", !isSalePermission) |>
  left_join(sr$uses, by = "pNbr") |>
  select(pNbr, use_nr,
    min_dosage, max_dosage, min_rate, max_rate, units_de,
    waiting_period, time_units_de, application_area_de)
print(boxer_uses)
```

```
# Show crop for use number 1
boxer_uses |>
  filter(use_nr == 1) |>
  left_join(sr$cultures, join_by(pNbr, use_nr)) |>
  select(use_nr, culture_de)

# Show target pests for use number 1
boxer_uses |>
  filter(use_nr == 1) |>
  left_join(sr$pests, join_by(pNbr, use_nr)) |>
  select(use_nr, pest_de)

# Show obligations for use number 1
boxer_uses |>
  filter(use_nr == 1) |>
  left_join(sr$obligations, join_by(pNbr, use_nr)) |>
  select(use_nr, sw_runoff_points, obligation_de) |>
  knitr::kable() |>
  print()

# Show application comments for use number 1
boxer_uses |>
  filter(use_nr == 1) |>
  left_join(sr$application_comments, join_by(pNbr, use_nr)) |>
  select(use_nr, application_comment_de)

# Illustrate 'obligations' indicating varying effects
sr$obligations |>
  filter(varying_effect) |>
  select(pNbr, use_nr, code, obligation_de)
```

`srppp_xml_clean_product_names` *Clean product names*

Description

Clean product names

Usage

```
srppp_xml_clean_product_names(names)
```

Arguments

`names`	Character vector of product names that should be cleaned from comments

Value

Character vector of cleaned names

`srppp_xml_define_use_numbers` *Define use identification numbers in an SRPPP read in from an XML file*

Description

Define use identification numbers in an SRPPP read in from an XML file

Usage

```
srppp_xml_define_use_numbers(srppp_xml = srppp_xml_get())
```

Arguments

`srppp_xml`	An object as returned by srppp_xml_get

Value

An object of the same class, with 'use_nr' added as an attribute of 'Indication' nodes.

Examples

```
try(srppp_xml_define_use_numbers())
```

`srppp_xml_get` *Read an XML version of the Swiss Register of Plant Protection Products*

Description

Read an XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_get(from, ...)

## S3 method for class '`NULL`'
srppp_xml_get(from, ...)

## S3 method for class 'character'
srppp_xml_get(from, ...)

srppp_xml_get_from_path(path, from)
```

Arguments

from	A specification of the way to retrieve the XML
...	Unused argument introduced to facilitate future extensions
path	A path to a zipped SRPPP XML file

Value

An object inheriting from 'srppp_xml', 'xml_document', 'xml_node'

Examples

```
# Try to get the current SRPPP as available from the FOAG website

srppp_cur <- try(srppp_xml_get())
```

---

srppp_xml_get_ingredients

Get ingredients for all registered products described in an XML version of the Swiss Register of Plant Protection Products

---

Description

Get ingredients for all registered products described in an XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_get_ingredients(srppp_xml = srppp_xml_get())
```

Arguments

srppp_xml	An object as returned by 'srppp_xml_get'

Value

A tibble::tibble containing a line for each ingredient of each W-Number

Examples

```
try(srppp_xml_get_ingredients())
```

srppp_xml_get_parallel_imports

Get Parallel Imports from an XML version of the Swiss Register of Plant Protection Products

Description

Get Parallel Imports from an XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_get_parallel_imports(srppp_xml = srppp_xml_get())
```

Arguments

srppp_xml	An object as returned by 'srppp_xml_get'

Value

A tibble::tibble with a row for each parallel import section in the XML file.

Examples

```
# Try to get current list of parallel_imports

try(srppp_xml_get_parallel_imports())
```

srppp_xml_get_products

Get Products from an XML version of the Swiss Register of Plant Protection Products

Description

Get Products from an XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_get_products(
  srppp_xml = srppp_xml_get(),
  verbose = TRUE,
  remove_duplicates = TRUE
)
```

Arguments

srppp_xml	An object as returned by 'srppp_xml_get'
verbose	Should we give some feedback?
remove_duplicates	Should duplicates based on wNbrs be removed? If set to 'TRUE', one of the two entries with identical wNbrs is removed, based on an investigation of background information carried out by the package authors. In all cases except for one, one of the product sections with duplicate wNbrs has information about an expiry of the registration, and the other doesn't. In these cases the registration without expiry is kept, and the expiring registration is discarded. In the remaining case (wNbr 5945), the second entry is selected, as it contains more indications which were apparently intended to be published as well.

Value

A tibble::tibble with a row for each product section in the XML file. An attribute 'duplicated_wNbrs' is also returned, containing duplicated W-Numbers, if applicable, or NULL.

Examples

```
# Try to get current list of products

try(srppp_xml_get_products())
```

---

srppp_xml_get_substances

Get substances from an XML version of the Swiss Register of Plant Protection Products

---

Description

Get substances from an XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_get_substances(srppp_xml = srppp_xml_get())
```

Arguments

srppp_xml	An object as returned by 'srppp_xml_get'

Value

A tibble::tibble containing primary keys, IUPAC names and substance names in German, French and Italian.

Examples

```
try(srppp_xml_get_substances())
```

`srppp_xml_get_uses` *Get uses ('indications') for all products described in an XML version of the Swiss Register of Plant Protection Products*

Description

Get uses ('indications') for all products described in an XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_get_uses(srppp_xml = srppp_xml_get())
```

Arguments

`srppp_xml`	An object as returned by srppp_xml_get with use numbers defined by srppp_xml_define_use_numbers

Value

A tibble::tibble of use definitions

Examples

```
srppp_xml <- try(srppp_xml_get())
if (!inherits(srppp_xml, "try-error")) {
  srppp_xml <- srppp_xml_define_use_numbers(srppp_xml)
  srppp_xml_get_uses(srppp_xml)
}
```

`srppp_xml_url` *URL of the XML version of the Swiss Register of Plant Protection Products*

Description

URL of the XML version of the Swiss Register of Plant Protection Products

Usage

```
srppp_xml_url
```

Format

length one character string

Examples

```
print(srppp_xml_url)
```

`units_convertible_to_g_per_ha` *Product application rate units convertible to grams active substance per hectare*

Description

Product application rate units convertible to grams active substance per hectare

Usage

```
units_convertible_to_g_per_ha
```

Format

An object of class `character` of length 7.

See Also

application_rate_g_per_ha

Examples

```
library(srppp)
library(dplyr)
# These are the convertible units
units_convertible_to_g_per_ha
```

Index