SOLUTIONS Substance Properties Database (SSPD) readme v1.8

Written by Rudy Schueder

Rudy.schueder@deltares.nl

March, 2017

Repository hosted @ https://github.com/dyprious/SSPD

0. Instructions for use

Please copy the folder SSPD (solutions substance properties database) somewhere on your computer. Also please ensure you have python 3.5 installed. This is necessary to read the unicode.

The database contains tables with information on substances and substance properties. It also contains data on how model output (like STREAM_EU or SIMPLE_TREAT) should interact with the properties in the database. The database is populated by importing all files within a certain folder and sorting it into tables.

To write model output, for which the only current options are STREAM_EU .inc format or SIMPLE_TREAT .csv format, a table of CAS numbers is created in the database based on substances found in the core data files. This table is then read by the the program, and a file for each substance is created. In this file are the model parameters for which substances exist in the database.

1. Current issues

1) Reference temperature should be a property in the database but it is currently not. It was not provided for all substances by all institutes. Currently it is assumed to be 25 deg C, as all the available values are 25 deg C. Right now it is prescribed to STREAM_EU using a list object called ref_temp in ..\py\search_list, and so can be manually edited
2) To ensure that units between the database and STREAM_EU are consistent, a manual check was conducted. The appropriate conversion factors were specified in column C of ../SSPD/database_properties/STREAM_EU_database_dictionary.csv. This is a somewhat manual check, and these values need to be adapted when the units of database properties change.

2. Database structure

Shown in is a schematic of the database structure and its 4 main tables with respect to a given model. For example STREAM_EU is shown here. # symbols denote the key of the table and arrows denote which column values are connected between tables. The table degredation_products, although present in this database build, is not populated and thus not included in the schematic.

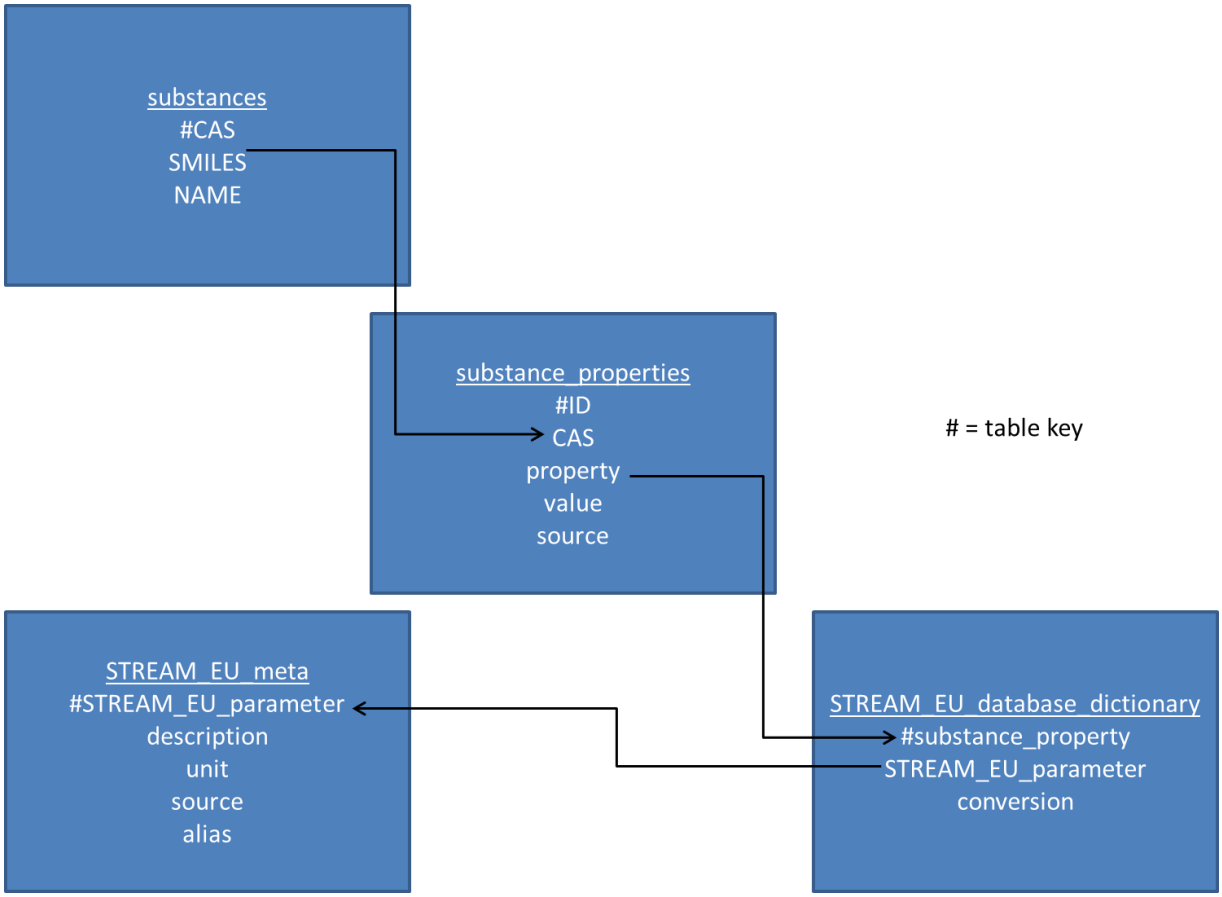

Figure 2.1: Schematic of the SSPD. The table degredation_products is not included because it is not yet used in the database.

3. Program folder structure

Within the SSPD folder there are 4 subfolders:

a. Database properties
   a. This contains csv files of metadata describing how the database should create model output. Provided is 'STREAM_EU_meta.csv', a file for specifying the contents of the STREAM_EU output in the form of include (.inc) files. All of the columns are relevant only for STREAM_EU. The file STREAM_EU_database_dictionary.csv defines the default search terms that will query the appropriate properties in the database along with the unit conversion value. Think of this as the 'link' between the STREAM_EU

parameters and the relevant properties in the database. The same hold for SIMPLE_TREAT files.

b. STREAM_EU_include_files
    a. This contains the STREAM_EU include files that are written as a result of running the python script. These are the outputs of the program.
c. SIMPLE_TREAT_inclue_files
    a. This contains the SIMPLE_TREAT include files that are written as a result of running the python script. These are the outputs of the program.
d. Properties_table
    a. In the folder ..\properties_table\overall is the comma delimited file containing all of the values fed to STREAM_EU .inc files for all of the substances.
e. Py
    a. This contains all of the program's scripts. The most important one is **make_database.py**, which is the script that is executed to run the program. The second important one is **search_list.py**. This is the file that allows you to have more control over which properties are put into the STREAM_EU include files (see section 7). The scripts **write_SIMPLE_TREAT_output.py** and **write_STREAM_EU_output.py** are used to create include files for these programs using the data in the database**.** The rest should generally not be used or edited.
f. Substance_properties
    a. Raw_data contains all of the files that you want to keep close at hand but do not want to be read by the program. It is simply used for storage and a dump of all data received.
    b. Source_properties contains the **tab delimited .txt or .csv** files that are used to populate the database. The program is somewhat flexible for the format of these files, but generally they must be in one of the following formats:

#	CAS	Name	SMILES	Property 1	Property 2	Property n…

CAS	Name	SMILES	Property 1	Property 2	Property n…

It is critical that the file be organized with one CAS number per row. The column organization is something the program is able to determine reasonably well on its own. However, if a file with column names that are not included in the list 'notprop' of the file **dynamic_entry.py**, then the upload from the file to the database will not work correctly. It is possible to have more than one layer (row) of headers.

4. Viewing the database

In the SSPD folder you will see a file called **substance_properties.db**. This is the SQLite database containing the substance properties. To view and interact with the database, please install the following software http://sqlitebrowser.org/

There is also a database created called substance_properties_manual.db. When first created, this database is an identical copy of substance_properties.db. It is simply created to

allow users to manually modify a recent database without needing to overwrite the one that was automatically written by the program.

5. Updating the database with new substances not for use in model output

To update the database, please follow these steps:

1) Place the files containing the properties you wish to add in the folder ../SSPD/substance_properties/source_properties
2) Open ../SSPD/py/make_database.py
3) Set the variable make_db = 1
4) Run **make_database.py**

6. Creating model output

In the case where the properties database is sufficiently up to date and you wish to produce model output, take the following steps (example is shown for STREAM_EU output):

1) Open ../SSPD/database_properties/STREAM_EU_database_dictionary.csv (or other dictionary)
2) Fill in columns A, B, and C and with the appropriate data. In Microsoft Excel, column A = substance_property name (in the database), column B = STREAM_EU parameter name, and C = unit correction factor.
3) The substance_property (column A) specifies which property in the database the adjacent STREAM_EU parameter (column B) is associated with. This term is essential, because it is the term that determines which properties, **by default (see section 7),** are amalgamated to produce the values written to the .inc file. For example, in the database for the single STREAM_EU parameter 'melting point', there exists the following properties:
    a. Melting Point (Gold and Ogle Method)
    b. Melting Point (Gold and Ogle Method)(min)
    c. Melting Point (Gold and Ogle Method)(max)
    d. Melting Point
    e. Melting Point (min)
    f. Melting Point (max)

    Therefore, the STREAM_EU parameter 'mp' is associated with all 6 of these melting points in the database table 'substance_properties', and thus 'mp' appears 6 times in the file 'STREAM_EU_database_dictionary.csv'. In this .csv, the user must specify all of the associated substance properties for a given STREAM_EU parameter to ensure that the database will return all of the desired variants of the property name. Of course there will be situations when you want to specify which single instance of the variants you want to use if you do not wish to use all of them. This will be elaborated on in section 7. However, if a parameter is to be written to the STREAM_EU include file as something other than '-9999', **a corresponding property name must be given** in 'STREAM_EU_database_dictionary.csv' for the writing algorithm to work properly, even if the property name will never be used as a search term.

4) The correction factor in column C is the value that the property value read from the database is multiplied by in order to achieve the units required by STREAM_EU ('-' means the inverse, followed by the multiplication factor. For example, the conversion factor -3600 means x value from the database becomes ln(2)/(3600x) in the .inc file). The value 3600 is chosen because half-lives are converted from the format x months, y days, z hours to z hours, and must be converted to 1/d.
5) Open ../SSPD/py/**write_STREAM_EU_output.py**
6) Run **write_STREAM_EU_output.py**

7. Writing desired model output with specified searches

The following is an example for STREAM_EU output. In the file ..\py**searchlist.py**, there is an object definition for 'method'. This is the object that defines how the program should decide which values from a database query will be written to the STREAM_EU .inc file. Method is an m x 1 dictionary where m is the number of substances in ../SSPD/database_properties/STREAM_EU_meta.csv. There are currently 3 valid default strings that can be put into method, along with the option to define your own string:

1) 'average' – only the search_term is used in the query. The average value of all valid query returns is used to produce the STREAM_EU value
2) 'min' – the search_term and the word 'min' are used in the query to produce the STREAM_EU value. When there are more than one results from this query, the average is taken
3) 'max' – the search_term and the word 'max' are used in the query to produce the STREAM_EU value. When there is more than one result from this query, the average is taken.
4) Specify – You can specify your own search string to specify which property in the database should be used for this STREAM_EU parameter. Including the 'method' object, there are 4 dictionaries that you can use to refine exactly which property is written to the STREAM_EU include files. If you put a string other than 'average', 'min', or 'max' into method, then the program will search for a property that contains the string you specified at that index of 'method'. For example, consider the following set-up for **search_list.py.**

```
method['kde'] = 'Ultimate Half Life Predicted'
method['kds'] = 'Ultimate Half Life Predicted'
method['kdw'] = 'Ultimate Half Life Predicted'

filespec1['kde'] = '301B'
filespec1['kds'] = '301B'
filespec1['kdw'] = '301B'

filespec2['kde'] = '301C'
filespec2['kds'] = '301C'
filespec2['kdw'] = '301C'

filespec3['kde'] = '301_F'
filespec3['kds'] = '301_F'
```

```
filespec3['kdw'] = '301_F'
```

As you can see, if you specify ‘Ultimate Half Life Predicted’ for property kds, kde, and kdw, then they will all take on the value of that specific property in the database.

The next three dictionaries are called filespec1, filespec2, and filespec3. These are optional additional search criteria for a specific STREAM_EU parameter. If something is specified here, it will look for the value that has ‘method[ii]’ in the name of the property and ‘filespec1[ii]’ in the name of the file from which the property came. If this returns an empty value, it then searches for ‘filespec2[ii]’, and if this is empty, it searches for ‘filespec3[ii]’. So, if in **search_list.py** I specify the following:

```
method['kde'] = 'Ultimate Half Life Predicted'
filespec1['kde'] = '301B'
filespec2['kde'] = '301C'
filespec3['kde'] = '301_F'
```

That means parameter kds in STREAM_EU will take on the parameter ‘Ultimate half-life predicted’. It will find three values of this name in the database, and will report the average if filespec1[‘kde’] is left empty. Because filespec1[‘kde’] is not empty, it will take the half-life from the 301B model. This is determined from the name of the file that supplied the parameter. If there is no entry for this or it is not a number, it will then search the 301C file and then the 301_F file. Filespec1 has priority over filespec2, and filespec2 has priority over filespec3.

8. Format of .inc file

The format of the .inc file follows the convention normally used in block #7 of the STREAM_EU input file. However, the comments provide additional information about the source of the numbers. The format of the text following the comment delimiter is as follows:

```
; parameter description          units          source
```