
SBU-Reporter Documentation

Release 0.4.0

B. F. van Beek

Sep 20, 2022

CONTENTS:

1	SBU-Reporter	1
1.1	Installation	1
1.2	Examples	1
2	SBU-Reporter API	3
2.1	sbu.dataframe	3
2.2	sbu.dataframe_postprocess	6
2.3	sbu.parse_yaml	8
2.4	sbu.plot	9
	Python Module Index	11
	Index	13

SBU-REPORTER

Tools for collection, formating and reporting SBU usage on the SURFsara HPC clusters.

More details are provided in the [documentation](#).

1.1 Installation

SBU-reporter can be installed as following:

- PyPi: `pip install git+https://github.com/BvB93/SBU-Reporter@v0.4.0 --upgrade`

1.2 Examples

```
get_sbu user_file.yaml --start=16-02-2021 --end=25-03-2022
```


SBU-REPORTER API

2.1 sbu.dataframe

A module which handles data parsing and DataFrame construction.

2.1.1 Index

<code>get_sbu(df, project[, start, end])</code>	Acquire the SBU usage for each account in the <code>pandas.DataFrame.index</code> .
<code>parse_accuse(project[, start, end])</code>	Gather SBU usage of a specific user account.
<code>get_date_range([start, end])</code>	Return a starting and ending date as two strings.
<code>construct_filename(prefix[, suffix])</code>	Construct a filename containing the current date.
<code>_get_datetimeindex(start, end)</code>	Create a Pandas DatetimeIndex from a start and end date.
<code>_parse_date(input_date[, default_day, ...])</code>	Parse any dates supplied to <code>get_date_range()</code> .
<code>_get_total_sbu_requested(df)</code>	Return the total number of requested SBUs.

2.1.2 API

`sbu.dataframe.get_sbu(df, project, start=None, end=None)`

Acquire the SBU usage for each account in the `pandas.DataFrame.index`.

The start and end of the reported interval can, optionally, be altered with **start** and **end**. Performs an inplace update of **df**, adding new columns to hold the SBU usage per month under the "Month" super-column. In addition, a single row and column is added ("sum") with SBU usage summed over the entire interval and over all users, respectively.

Parameters

- **df** (`pandas.DataFrame`) – A Pandas DataFrame with usernames and information, constructed by `yaml_to_pandas()`. `pandas.DataFrame.columns` and `pandas.DataFrame.index` should be instances of `pandas.MultiIndex` and `pandas.Index`, respectively. User accounts are expected to be stored in `pandas.DataFrame.index`. SBU usage (including the sum) is stored in the "Month" super-column.
- **start** (`int` or `str`, optional) – Optional: The starting year of the interval. Defaults to the current year if None.
- **end** (`str` or `int`, optional) – Optional: The final year of the interval. Defaults to current year + 1 if None.

- **project** (*str*, optional) – Optional: The project code of the project of interest. If not *None*, only SBUs expended under this project are considered.

Return type*None*

```
sbu.dataframe.parse_accuse(project, start=None, end=None)
```

Gather SBU usage of a specific user account.

The bash command `accuse` is used for gathering SBU usage along an interval defined by **start** and **end**. Results are collected and returned in a Pandas DataFrame.

Parameters

- **project** (*str*) – The project code of the project of interest.
- **start** (*str*) – The starting date of the interval. Accepts dates formatted as YYYY, MM-YYYY or DD-MM-YYYY.
- **end** (*str*) – The final date of the interval. Accepts dates formatted as YYYY, MM-YYYY or DD-MM-YYYY.

Returns

The SBU usage of **user** over a specified period.

Return type*pandas.DataFrame*

```
sbu.dataframe.get_date_range(start=None, end=None)
```

Return a starting and ending date as two strings.

Parameters

- **start** (*int* or *str*, optional) – The starting year of the interval. Accepts dates formatted as YYYY, MM-YYYY or DD-MM-YYYY. Defaults to the current year if *None*.
- **end** (*str* or *int*, optional) – The final year of the interval. Accepts dates formatted as YYYY, MM-YYYY or DD-MM-YYYY. Defaults to the current year + 1 if *None*.

Returns

A tuple with the start and end data, formatted as strings. Dates are formatted as DD-MM-YYYY.

Return type*tuple [str, str]*

```
sbu.dataframe.construct_filename(prefix, suffix='csv')
```

Construct a filename containing the current date.

Examples

```
>>> filename = construct_filename('my_file', '.txt')
>>> print(filename)
'my_file_31_May_2019.txt'
```

Parameters

- **prefix** (*str*) – A prefix for the to-be returned filename. The current date will be appended to this prefix.
- **suffix** (*str*, optional) – An optional suffix of the to be returned filename. No suffix will be attached if *None*.

Returns

A filename consisting of **prefix**, the current date and **suffix**.

Return type

`str`

`sbu.dataframe._get_datetimeindex(start, end)`

Create a Pandas DatetimeIndex from a start and end date.

Parameters

- **start** (`str`) – The start of the interval. Accepts dates formatted as DD-MM-YYYY.
- **end** (`str`) – The end of the interval. Accepts dates formatted as DD-MM-YYYY.

Returns

A DatetimeIndex starting from **sy** and ending on **ey**.

Return type

`pandas.DatetimeIndex`

`sbu.dataframe._parse_date(input_date, default_day='01', default_month='01', default_year=None)`

Parse any dates supplied to `get_date_range()`.

Parameters

- **input_date** (`str`, `int` or `None`) – The to-be parsed date. Allowed types and values are:
 - `None`: Defaults to the first day of the current year and month.
 - `int`: A year (e.g. 2019).
 - `str`: A date in YYYY, MM-YYYY or DD-MM-YYYY format (e.g. "22-10-2018").
- **default_month** (`str`) – The default month if a month is not provided in **input_date**. Expects a month in MM format.
- **default_year** (`str`, optional) – Optional: The default year if a year is not provided in **input_date**. Expects a year in YYYY format. Defaults to the current year if `None`.

Returns

A string, constructed from **input_date**, representing a date in DD-MM-YYYY format.

Return type

`str`

Raises

- **ValueError** – Raised if **input_date** is provided as string and contains more than 2 dashes.
- **TypeError** – Raised if **input_date** is neither `None`, a string nor an integer.

`sbu.dataframe._get_total_sbu_requested(df)`

Return the total number of requested SBUs.

Return type

`float`

2.2 sbu.dataframe_postprocess

A module for creating new dataframes from the SBU-containing dataframe.

2.2.1 Index

<code>get_sbu_per_project(df)</code>	Construct a new Pandas DataFrame with SBU usage per project.
<code>get_agregated_sbu(df)</code>	Calculate the SBU accumulated over all months in the "Month" super-column.
<code>get_percentage_sbu(df)</code>	Calculate the % accumulated SBU usage per project.
<code>_get_active_name(df, index)</code>	Return a tuple with the names of all active users.

2.2.2 API

`sbu.dataframe_postprocess.get_sbu_per_project(df)`

Construct a new Pandas DataFrame with SBU usage per project.

Parameters

df (`pandas.DataFrame`) – A Pandas DataFrame with SBU usage per username, constructed by `get_sbu()`. `pandas.DataFrame.columns` and `pandas.DataFrame.index` should be instances of `pandas.MultiIndex` and `pandas.Index`, respectively.

Returns

A new Pandas DataFrame holding the SBU usage per project (*i.e.* `df [project]`).

Return type

`pandas.DataFrame`

`sbu.dataframe_postprocess.get_agregated_sbu(df)`

Calculate the SBU accumulated over all months in the "Month" super-column.

Examples

Considering the following DataFrame as input:

```
>>> print(df['Month'])
          2019-01  2019-02  2019-03
username
Donald Duck    1000.0   1500.0    750.0
Scrooge McDuck 1000.0    500.0    250.0
Mickey Mouse   1000.0   5000.0   4000.0
```

Which will be accumulated along each column in the following manner:

```
>>> df_new = get_agregated_sbu(df)
>>> print(df_new['Month'])
          2019-01  2019-02  2019-03
username
Donald Duck    1000.0   2500.0   3250.0
Scrooge McDuck 1000.0   1500.0   1750.0
Mickey Mouse   1000.0   6000.0  10000.0
```

Parameters

df (`pandas.DataFrame`) – A Pandas DataFrame with SBU usage per project, constructed by `get_sbu_per_project()`. `pandas.DataFrame.columns` and `pandas.DataFrame.index` should be instances of `pandas.MultiIndex` and `pandas.Index`, respectively.

Returns

A new Pandas DataFrame with SBU usage accumulated over all columns in the "Month" super-column.

Return type

`pandas.DataFrame`

```
sbu.dataframe_postprocess.get_percentage_sbu(df)
```

Calculate the % accumulated SBU usage per project.

The column storing the requested amount of SBUs can be defined in the global variable `_GLOBVAR["SBU_REQUESTED"]` (default value: ("info", "SBU requested")).

Examples

Considering the following DataFrame with accumulated SBUs as input:

```
>>> print(df)
```

	info	Month		
	SBU requested	2019-01	2019-02	2019-03
username				
Donald Duck	3250.0	1000.0	2500.0	3250.0
Scrooge McDuck	5000.0	1000.0	1500.0	1750.0
Mickey Mouse	5000.0	1000.0	6000.0	10000.0

Which will result in the following SBU usage:

```
>>> df_new = get_percentage_sbu(df)
>>> print(df_new['Month'])
```

	2019-01	2019-02	2019-03
username			
Donald Duck	0.31	0.77	1.00
Scrooge McDuck	0.20	0.30	0.35
Mickey Mouse	0.20	1.20	2.00

Parameters

df (`pandas.DataFrame`) – A Pandas DataFrame with the accumulated SBU usage per project, constructed by `get_agregated_sbu()`. `pandas.DataFrame.columns` and `pandas.DataFrame.index` should be instances of `pandas.MultiIndex` and `pandas.Index`, respectively.

Returns

A new Pandas DataFrame with % SBU usage accumulated over all columns in the "Month" super-column.

Return type

`pandas.DataFrame`

```
sbu.dataframe_postprocess._get_active_name(df, index)
```

Return a tuple with the names of all active users.

Return type
tuple

2.3 sbu.parse_yaml

A module for parsing and validating the .yaml input.

2.3.1 Index

<code>yaml_to_pandas(filename)</code>	Create a Pandas DataFrame out of a .yaml file.
<code>validate_usernames(df)</code>	Validate that all users belonging to an account are available in the .yaml input file.

2.3.2 API

```
sbu.parse_yaml.yaml_to_pandas(filename)
```

Create a Pandas DataFrame out of a .yaml file.

Examples

Example yaml input:

```
__project__: BlaBla
A:
  description: Example project
  PI: Walt Disney
  SBU requested: 1000
  users:
    user1: Donald Duck
    user2: Scrooge McDuck
    user3: Mickey Mouse
```

Example output:

```
>>> df, project = yaml_to_pandas(filename)
>>> print(df)
      info      name  ... SBU requested      PI
username
user1      A  Donald Duck  ...      1000.0  Walt Disney
user2      A  Scrooge McDuck  ...      1000.0  Walt Disney
user3      A   Mickey Mouse  ...      1000.0  Walt Disney

>>> print(project)
BlaBla
```

Parameters

filename (`str`) – The path+filename to the .yaml file.

Returns

A Pandas DataFrame and project name constructed from **filename**. Columns and rows are instances of `pandas.MultiIndex` and `pandas.Index`, respectively. All retrieved .yaml data is stored under the "info" super-column. The project name will be `None` if the `__project__` key is absent from the .yaml file

Return type

`pandas.DataFrame` & `str`, optional

`sbu.parse_yaml.validate_usernames(df)`

Validate that all users belonging to an account are available in the .yaml input file.

Raises a `KeyError` If one or more usernames printed by the `accinfo` comand are absent from **df**.

Parameters

df (`pandas.DataFrame`) – A DataFrame, produced by `yaml_to_pandas()`, containing user accounts. `pandas.DataFrame.columns` and `pandas.DataFrame.index` should be instances of `pandas.MultiIndex` and `pandas.Index`, respectively. User accounts are expected to be stored in `pandas.DataFrame.index`.

Raises

ValueError – Raised if one or more users reported by the `accinfo` command are absent from **df** or *vice versa*.

Return type

`None`

2.4 sbu.plot

A module for handling data plotting.

2.4.1 Index

2.4.2 API

PYTHON MODULE INDEX

S

`sbu.dataframe`, [3](#)
`sbu.dataframe_postprocess`, [5](#)
`sbu.parse_yaml`, [8](#)
`sbu.plot_fig`, [9](#)

Symbols

`_get_active_name()` (in module `sbu.dataframe_postprocess`), 7
`_get_datetimeindex()` (in module `sbu.dataframe`), 5
`_get_total_sbu_requested()` (in module `sbu.dataframe`), 5
`_parse_date()` (in module `sbu.dataframe`), 5

C

`construct_filename()` (in module `sbu.dataframe`), 4

G

`get_agregated_sbu()` (in module `sbu.dataframe_postprocess`), 6
`get_date_range()` (in module `sbu.dataframe`), 4
`get_percentage_sbu()` (in module `sbu.dataframe_postprocess`), 7
`get_sbu()` (in module `sbu.dataframe`), 3
`get_sbu_per_project()` (in module `sbu.dataframe_postprocess`), 6

M

module
`sbu.dataframe`, 3
`sbu.dataframe_postprocess`, 5
`sbu.parse_yaml`, 8
`sbu.plot_fig`, 9

P

`parse_accuse()` (in module `sbu.dataframe`), 4

S

`sbu.dataframe`
module, 3
`sbu.dataframe_postprocess`
module, 5
`sbu.parse_yaml`
module, 8
`sbu.plot_fig`
module, 9

V

`validate_usernames()` (in module `sbu.parse_yaml`), 9

Y

`yaml_to_pandas()` (in module `sbu.parse_yaml`), 8