HBP Validation Framework - Python Client Documentation

Release 0.5.6

Shailesh Appukuttan

Aug 01, 2019

Contents

A Python package for working with the Human Brain Project Model Validation Framework.

Andrew Davison and Shailesh Appukuttan, CNRS, 2017

CHAPTER 1

Quick Overview

We discuss here some of the terms used in this documentation.

Model A Model or Model description consists of all the information pertaining to a model excluding details of the source code (i.e. implementation). The model would specify metadata describing the model type and its domain of utility. The source code is specified via the model instance (see below).

Model Instance This defines a particular version of a model by specifying the location of the source code for the model. A model may have multiple versions (model instances) which could vary, for example, in values of their biophysical parameters. Improvements and updates to a model would be considered as different versions (instances) of that particular model.

Test A Test or Test definition consists of all the information pertaining to a test excluding details of the source code (i.e. implementation). The test would specify metadata defining its domain of utility along with other info such as the type of data it handles and the type of score it generates. The source code is specified via the test instance (see below).

Test Instance This defines a particular version of a test by specifying the location of the source code for executing the test. A test may have multiple versions (test instances) which could vary, for example, in the way the simulation is setup or how the score is evaluated. Improvements in the test code would be considered as different versions (instances) of that particular test.

sciunit A Python package that handles testing of models. For more, see: https://github.com/scidash/sciunit

Result The outcome of testing a specific model instance with a specific test instance. The result would consist of a score, and possibly additionally output files generated by the test.

More detailed tutorials will be published soon.

For any queries, you can contact:

- Andrew Davison: `andrew.davison@unic.cnrs-gif.fr`
- Shailesh Appukuttan: `shailesh.appukuttan@unic.cnrs-gif.fr`

CHAPTER 2

General Info

- From the above descriptions, it can be identified that *running a particular test for a model* under the validation framework is more accurately described as the *running of a specific test instance for a specific model instance*.
- When running a test, the test metadata and test instance info is typically retrieved from the validation framework. This involves authenticating your HBP login credentials.
- The model being tested can be registered on the *Model Catalog* beforehand, or asked to be registered automatically after the test is complete, just before registering the result on the validation framework.
- Registration of the model and its test results also require authenticating your HBP login credentials.
- It should be noted that an HBP account can be created even by non-HBP users. For more information, please visit: https://services.humanbrainproject.eu/oidc/account/request
- Collabs on the HBP Collaboratory can be either public or private. Public Collabs can be accessed by all registered users, whereas private Collabs require the user to be granted permission for access.
- The *Model Catalog* and the *Validation Framework* apps can be added to any Collab. A Collab may have multiple instances of these apps. The apps require to be *configured* by setting the provided filters appropriately before they can be used. These filters restrict the type of data displayed in that particular instance of the app.
- All tests are public, i.e. every test registered on the *Validation Framework* can be seen by all users.
- Models are created inside specific Collab instances of the *Model Catalog* app. The particular app inside which a model was created is termed its *host app*. Similarly, the Collab containing the *host app* is termed the *host Collab*.
- Models can be set as public/private. If public, the model and its associated results are available to all users. If private, it can only be seen by users who have access to the *host Collab*. See table below for summary of access privileges.
- No information can be deleted from the *Model Catalog* and *Validation Framework* apps. In future, an option to *hide* data would be implemented. This would offer users functionality similar to deleting, but with the data being retained in the database back-end.
- Models, model instances, tests and test instances can be edited as long as there are no results associated with them. Results can never be edited!

CHAPTER 3

Regarding HBP Authentication

The Python Client for the Validation Framework attempts to simplify the HBP authentication process. It does this as follows:

On first use, the users have the following options (in order of priority):

1. Setting an environment variable named `HBP_PASS` with your HBP password. On Linux, this can be done as:

   ```
   export HBP_PASS='putyourpasswordhere'
   ```

   Environment variables set like this are only stored temporally. When you exit the running instance of bash by exiting the terminal, they get discarded. To save this permanentally, write the above command into *~/.bashrc* or *~/.profile* (you might need to reload these files by, for example, `source ~/.bashrc`)

2. Enter your HBP password when prompted by the Python Client.

Once you do either of the two, the Python Client will save the retrieved token locally on your system. Henceforth, this token would be used for all subsequent requests that require authentication. This approach has been found to significantly speed-up the processing of the requests. If the authentication token expires, or is found invalid, then the user would again be give the above two options.

CHAPTER 4

TestLibrary

class hbp_validation_framework.**TestLibrary**(*username=None*, *password=None*, *environment='production'*)

Client for the HBP Validation Test library.

The TestLibrary client manages all actions pertaining to tests and results. The following actions can be performed:

Action	Method
Get test definition	*get_test_definition()*
Get test as Python (sciunit) class	*get_validation_test()*
List test definitions	*list_tests()*
Add new test definition	*add_test()*
Edit test definition	*edit_test()*
Get test instances	*get_test_instance()*
List test instances	*list_test_instances()*
Add new test instance	*add_test_instance()*
Edit test instance	*edit_test_instance()*
Get valid attribute values	*get_attribute_options()*
Get test result	*get_result()*
List test results	*list_results()*
Register test result	*register_result()*

Parameters

- **username** (*string*) – Your HBP Collaboratory username. Not needed in Jupyter notebooks within the HBP Collaboratory.
- **password** (*string, optional*) – Your HBP Collaboratory password; advisable to not enter as plaintext. If left empty, you would be prompted for password at run time (safer). Not needed in Jupyter notebooks within the HBP Collaboratory.
- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system,

which is appropriate for most users. When set to *dev*, it uses the *development* system. Other environments, if required, should be defined inside a json file named *config.json* in the working directory. Example:

```
{
  "prod": { "url": "https://validation-v1.brainsimulation.eu", "client_id": "3ae21f28-
    0302-4d28-8581-15853ad6107d"
  }, "dev_test": {
    "url": "https://localhost:8000", "client_id": "90c719e0-29ce-43a2-9c53-
    15cb314c2d0b", "verify_ssl": false
  }
}
```

Examples

Instantiate an instance of the TestLibrary class

```
>>> test_library = TestLibrary(hbp_username)
```

get_test_definition(*test_path=''*, *test_id=''*, *alias=''*)

Retrieve a specific test definition.

A specific test definition can be retrieved from the test library in the following ways (in order of priority):

1. load from a local JSON file specified via *test_path*
2. specify the *test_id*
3. specify the *alias* (of the test)

Parameters

- **test_path** (*string*) – Location of local JSON file with test definition.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.

Note: Also see: *get_validation_test()*

Returns Information about the test.

Return type dict

Examples

```
>>> test = test_library.get_test_definition("/home/shailesh/Work/dummy_test.
↪json")
>>> test = test_library.get_test_definition(test_id="7b63f87b-d709-4194-bae1-
↪15329daf3dec")
>>> test = test_library.get_test_definition(alias="CDT-6")
```

get_validation_test(*test_path=''*, *instance_path=''*, *instance_id=''*, *test_id=''*, *alias=''*, *version=''*, ***params*)

Retrieve a specific test instance as a Python class (sciunit.Test instance).

A specific test definition can be specified in the following ways (in order of priority):

1. load from a local JSON file specified via *test_path* and *instance_path*
2. specify *instance_id* corresponding to test instance in test library
3. specify *test_id* and *version*
4. specify *alias* (of the test) and *version* Note: for (3) and (4) above, if *version* is not specified,

   then the latest test version is retrieved

Parameters

- **test_path** (*string*) – Location of local JSON file with test definition.
- **instance_path** (*string*) – Location of local JSON file with test instance metadata.
- **instance_id** (*UUID*) – System generated unique identifier associated with test instance.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.
- **version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.
- ****params** – Additional keyword arguments to be passed to the Test constructor.

Note: To confirm the priority of parameters for specifying tests and instances, see *get_test_definition()* and *get_test_instance()*

Returns Returns a `sciunit.Test` instance.

Return type sciunit.Test

Examples

```
>>> test = test_library.get_validation_test(alias="CDT-6", instance_id=
↪"36a1960e-3e1f-4c3c-a3b6-d94e6754da1b")
```

list_tests(***filters*)

Retrieve a list of test definitions satisfying specified filters.

The filters may specify one or more attributes that belong to a test definition. The following test attributes can be specified:

- name

- alias
- author
- species
- age
- brain_region
- cell_type
- data_modality
- test_type
- score_type
- model_scope
- abstraction_level
- data_type
- publication

Parameters ****filters** (*variable length keyword arguments*) – To be used to filter test definitions from the test library.

Returns List of model descriptions satisfying specified filters.

Return type list

Examples

```
>>> tests = test_library.list_tests()
>>> tests = test_library.list_tests(test_type="single cell activity")
>>> tests = test_library.list_tests(test_type="single cell activity", cell_
↪type="Pyramidal Cell")
```

add_test(*name=''*, *alias=None*, *version=''*, *author=''*, *species=''*, *age=''*, *brain_region=''*, *cell_type=''*, *data_modality=''*, *test_type=''*, *score_type=''*, *protocol=''*, *data_location=''*, *data_type=''*, *publication=''*, *repository=''*, *path=''*)

Register a new test on the test library.

This allows you to add a new test to the test library. A test instance (version) needs to be specified when registering a new test.

Parameters

- **name** (*string*) – Name of the test definition to be created.
- **alias** (*string, optional*) – User-assigned unique identifier to be associated with test definition.
- **version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.
- **author** (*string*) – Name of person creating the test.
- **species** (*string*) – The species from which the data was collected.
- **age** (*string*) – The age of the specimen.
- **brain_region** (*string*) – The brain region being targeted in the test.

- **cell_type** (*string*) – The type of cell being examined.
- **data_modality** (*string*) – Specifies the type of observation used in the test.
- **test_type** (*string*) – Specifies the type of the test.
- **score_type** (*string*) – The type of score produced by the test.
- **protocol** (*string*) – Experimental protocol involved in obtaining reference data.
- **data_location** (*string*) – URL of file containing reference data (observation).
- **data_type** (*string*) – The type of reference data (observation).
- **publication** (*string*) – Publication or comment (e.g. "Unpublished") to be associated with observation.
- **repository** (*string*) – URL of Python package repository (e.g. GitHub).
- **path** (*string*) – Python path (not filesystem path) to test source code within Python package.

Returns UUID of the test instance that has been created.

Return type UUID

Examples

```
>>> test = test_library.add_test(name="Cell Density Test", alias="", version=
↪"1.0", author="Shailesh Appukuttan",
                         species="Mouse (Mus musculus)", age="TBD", brain_
↪region="Hippocampus", cell_type="Other",
                         data_modality="electron microscopy", test_type=
↪"network structure", score_type="Other", protocol="Later",
                         data_location="collab://Validation Framework/
↪observations/test_data/cell_density_Halasy_1996.json",
                         data_type="Mean, SD", publication="Halasy et al., 1996
↪",
                         repository="https://github.com/appukuttan-shailesh/
↪morphounit.git", path="morphounit.tests.CellDensityTest")
```

edit_test(*name=None*, *test_id=''*, *alias=None*, *author=None*, *species=None*, *age=None*, *brain_region=None*, *cell_type=None*, *data_modality=None*, *test_type=None*, *score_type=None*, *protocol=None*, *data_location=None*, *data_type=None*, *publication=None*)

Edit an existing test in the test library.

To update an existing test, the *test_id* must be provided. Any of the other parameters may be updated. Only the parameters being updated need to be specified.

Parameters

- **name** (*string*) – Name of the test definition.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string, optional*) – User-assigned unique identifier to be associated with test definition.
- **author** (*string*) – Name of person who created the test.
- **species** (*string*) – The species from which the data was collected.
- **age** (*string*) – The age of the specimen.

- **brain_region** (*string*) – The brain region being targeted in the test.
- **cell_type** (*string*) – The type of cell being examined.
- **data_modality** (*string*) – Specifies the type of observation used in the test.
- **test_type** (*string*) – Specifies the type of the test.
- **score_type** (*string*) – The type of score produced by the test.
- **protocol** (*string*) – Experimental protocol involved in obtaining reference data.
- **data_location** (*string*) – URL of file containing reference data (observation).
- **data_type** (*string*) – The type of reference data (observation).
- **publication** (*string*) – Publication or comment (e.g. "Unpublished") to be associated with observation.

Note: Test instances cannot be edited here. This has to be done using *edit_test_instance()*

Returns (Verify!) UUID of the test instance that has been edited.

Return type UUID

Examples

test = test_library.edit_test(name="Cell Density Test", test_id="7b63f87b-d709-4194-bae1-15329daf3dec", alias="CD
species="Mouse (Mus musculus)", brain_region="Hippocampus", cell_type="Other", age="TBD", data_modality="electron microscopy", test_type="network structure", score_type="Other", protocol="To be filled sometime later", data_location="collab://Validation Framework/observations/test_data/cell_density_Halasy_1996.json", data_type="Mean, SD")

delete_test(*test_id=''*, *alias=''*)

ONLY FOR SUPERUSERS: Delete a specific test definition by its test_id or alias.

A specific test definition can be deleted from the test library, along with all associated test instances, in the following ways (in order of priority):

1. specify the *test_id*
2. specify the *alias* (of the test)

Parameters

- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.

Note:

- This feature is only for superusers!

Examples

```
>>> test_library.delete_test(test_id="8c7cb9f6-e380-452c-9e98-e77254b088c5")
>>> test_library.delete_test(alias="B1")
```

get_test_instance(*instance_path=''*, *instance_id=''*, *test_id=''*, *alias=''*, *version=''*)

Retrieve a specific test instance definition from the test library.

A specific test instance can be retrieved in the following ways (in order of priority):

1. load from a local JSON file specified via *instance_path*
2. specify *instance_id* corresponding to test instance in test library
3. specify *test_id* and *version*
4. specify *alias* (of the test) and *version* Note: for (3) and (4) above, if *version* is not specified,

   then the latest test version is retrieved

Parameters

- **instance_path** (*string*) – Location of local JSON file with test instance metadata.
- **instance_id** (*UUID*) – System generated unique identifier associated with test instance.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.
- **version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.

Returns Information about the test instance.

Return type dict

Examples

```
>>> test_instance = test_library.get_test_instance(test_id="7b63f87b-d709-
↪4194-bae1-15329daf3dec", version="1.0")
>>> test_instance = test_library.get_test_instance(test_id="7b63f87b-d709-
↪4194-bae1-15329daf3dec")
```

list_test_instances(*instance_path=''*, *test_id=''*, *alias=''*)

Retrieve list of test instances belonging to a specified test.

This can be retrieved in the following ways (in order of priority):

1. load from a local JSON file specified via *instance_path*
2. specify *test_id*
3. specify *alias* (of the test)

Parameters

- **instance_path** (*string*) – Location of local JSON file with test instance metadata.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.

Returns Information about the test instances.

Return type dict[]

Examples

```
>>> test_instances = test_library.list_test_instances(test_id="8b63f87b-d709-
↪4194-bae1-15329daf3dec")
```

add_test_instance(*test_id=''*, *alias=''*, *repository=''*, *path=''*, *version=''*, *description=''*, *parameters=''*)

Register a new test instance.

This allows to add a new instance to an existing test in the test library. The *test_id* needs to be specified as input parameter.

Parameters

- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.
- **repository** (*string*) – URL of Python package repository (e.g. github).
- **path** (*string*) – Python path (not filesystem path) to test source code within Python package.
- **version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.
- **description** (*string, optional*) – Text describing this specific test instance.
- **parameters** (*string, optional*) – Any additional parameters to be submitted to test, or used by it, at runtime.

Returns UUID of the test instance that has been created.

Return type UUID

Note:

- *alias* is not currently implemented in the API; kept for future use.
- TODO: Either test_id or alias needs to be provided, with test_id taking precedence over alias.

Examples

```
>>> response = test_library.add_test_instance(test_id="7b63f87b-d709-4194-
↪bae1-15329daf3dec",
                                              repository="https://github.com/appukuttan-
↪shailesh/morphounit.git",
                                              path="morphounit.tests.CellDensityTest",
                                              version="3.0")
```

edit_test_instance(*instance_id=''*, *test_id=''*, *alias=''*, *repository=None*, *path=None*, *version=None*, *description=None*, *parameters=None*)

Edit an existing test instance.

This allows to edit an instance of an existing test in the test library. The test instance can be specified in the following ways (in order of priority):

1. specify *instance_id* corresponding to test instance in test library
2. specify *test_id* and *version*
3. specify *alias* (of the test) and *version*

Only the parameters being updated need to be specified. You cannot edit the test *version* in the latter two cases. To do so, you must employ the first option above. You can retrieve the *instance_id* via *get_test_instance()*

Parameters

- **instance_id** (*UUID*) – System generated unique identifier associated with test instance.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.
- **repository** (*string*) – URL of Python package repository (e.g. github).
- **path** (*string*) – Python path (not filesystem path) to test source code within Python package.
- **version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.
- **description** (*string, optional*) – Text describing this specific test instance.
- **parameters** (*string, optional*) – Any additional parameters to be submitted to test, or used by it, at runtime.

Returns UUID of the test instance that has was edited.

Return type UUID

Examples

```
>>> response = test_library.edit_test_instance(test_id="7b63f87b-d709-4194-
↪bae1-15329daf3dec",
                                repository="https://github.com/appukuttan-
↪shailesh/morphounit.git",
                                path="morphounit.tests.CellDensityTest",
                                version="4.0")
```

delete_test_instance(*instance_id=''*, *test_id=''*, *alias=''*, *version=''*)
ONLY FOR SUPERUSERS: Delete an existing test instance.

This allows to delete an instance of an existing test in the test library. The test instance can be specified in the following ways (in order of priority):

1. specify *instance_id* corresponding to test instance in test library
2. specify *test_id* and *version*
3. specify *alias* (of the test) and *version*

Parameters

- **instance_id** (*UUID*) – System generated unique identifier associated with test instance.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **alias** (*string*) – User-assigned unique identifier associated with test definition.
- **version** (*string*) – User-assigned unique identifier associated with test instance.

Note:

- This feature is only for superusers!

Examples

```
>>> test_library.delete_model_instance(test_id="8c7cb9f6-e380-452c-9e98-
↪e77254b088c5")
>>> test_library.delete_model_instance(alias="B1", version="1.0")
```

get_attribute_options(*param=''*)

Retrieve valid values for test attributes.

Will return the list of valid values (where applicable) for various test attributes. The following test attributes can be specified:

- cell_type
- test_type
- score_type
- brain_region
- data_modalities
- species

If an attribute is specified, then only values that correspond to it will be returned, else values for all attributes are returned.

Parameters **param** (*string, optional*) – Attribute of interest

Returns Dictionary with key(s) as attribute(s), and value(s) as list of valid options.

Return type dict

Examples

```
>>> data = test_library.get_attribute_options()
>>> data = test_library.get_attribute_options("cell_type")
```

get_result(*result_id=''*, *order=''*)

Retrieve a test result.

This allows to retrieve the test result score and other related information. The *result_id* needs to be specified as input parameter.

Parameters

- **result_id** (*UUID*) – System generated unique identifier associated with result.
- **order** (*string, optional*) – Determines how the result should be structured. Valid values are "test", "model" or "". Default is "" and provides concise result summary.

Returns Information about the result retrieved.

Return type dict

Examples

```
>>> result = test_library.get_result(result_id="901ac0f3-2557-4ae3-bb2b-
↪37617312da09")
>>> result = test_library.get_result(result_id="901ac0f3-2557-4ae3-bb2b-
↪37617312da09", order="test")
```

list_results (*order=''*, ***filters*)

Retrieve test results satisfying specified filters.

This allows to retrieve a list of test results with their scores and other related information.

Parameters

- **order** (*string, optional*) – Determines how the result should be structured. Valid values are "test", "model" or "". Default is "" and provides concise result summary.
- ****filters** (*variable length keyword arguments*) – To be used to filter the results metadata.

Returns Information about the results retrieved.

Return type dict

Examples

```
>>> results = test_library.list_results()
>>> results = test_library.list_results(order="test", test_id="7b63f87b-d709-
↪4194-bae1-15329daf3dec")
>>> results = test_library.list_results(id="901ac0f3-2557-4ae3-bb2b-
↪37617312da09")
>>> results = test_library.list_results(model_version_id="f32776c7-658f-462f-
↪a944-1daf8765ec97", order="test")
```

register_result (*test_result*, *data_store=None*, *project=None*)

Register test result with HBP Validation Results Service.

The score of a test, along with related output data such as figures, can be registered on the validation framework.

Parameters

- **test_result** (sciunit.Score) – a sciunit.Score instance returned by *test.judge(model)*
- **data_store** (DataStore) – a DataStore instance, for uploading related data generated by the test run, e.g. figures.
- **project** (*int*) – Numeric input specifying the Collab ID, e.g. 8123. This is used to indicate the Collab where results should be saved.

Note: Source code for this method still contains comments/suggestions from previous client. To be removed or implemented.

Returns UUID of the test result that has been created.

Return type UUID

Examples

```
>>> score = test.judge(model)
>>> response = test_library.register_result(test_result=score)
```

delete_result(*result_id=''*)

ONLY FOR SUPERUSERS: Delete a result on the validation framework.

This allows to delete an existing result info on the validation framework. The *result_id* needs to be specified as input parameter.

Parameters **result_id** (*UUID*) – System generated unique identifier associated with result.

Note:

- This feature is only for superusers!

Examples

```
>>> model_catalog.delete_result(result_id="2b45e7d4-a7a1-4a31-a287-
↪aee7072e3e75")
```

CHAPTER 5

ModelCatalog

class hbp_validation_framework.**ModelCatalog**(*username=None*, *password=None*, *environment='production'*)

Client for the HBP Model Catalog.

The ModelCatalog client manages all actions pertaining to models. The following actions can be performed:

Action	Method
Get model description	*get_model()*
List model descriptions	*list_models()*
Register new model description	*register_model()*
Edit model description	*edit_model()*
Get valid attribute values	*get_attribute_options()*
Get model instance	*get_model_instance()*
Download model instance	*download_model_instance()*
List model instances	*list_model_instances()*
Add new model instance	*add_model_instance()*
Find model instance; else add	*find_model_instance_else_add()*
Edit existing model instance	*edit_model_instance()*
Get figure from model description	*get_model_image()*
List figures from model description	*list_model_images()*
Add figure to model description	*add_model_image()*
Edit existing figure metadata	*edit_model_image()*

Parameters

- **username** (*string*) – Your HBP Collaboratory username. Not needed in Jupyter notebooks within the HBP Collaboratory.
- **password** (*string, optional*) – Your HBP Collaboratory password; advisable to not enter as plaintext. If left empty, you would be prompted for password at run time (safer). Not needed in Jupyter notebooks within the HBP Collaboratory.

- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system, which is appropriate for most users. When set to *dev*, it uses the *development* system. Other environments, if required, should be defined inside a json file named *config.json* in the working directory. Example:

```
{
  "prod": { "url": "https://validation-v1.brainsimulation.eu", "client_id": "3ae21f28-0302-4d28-8581-15853ad6107d"
  }, "dev_test": {
    "url": "https://localhost:8000", "client_id": "90c719e0-29ce-43a2-9c53-15cb314c2d0b", "verify_ssl": false
  }
}
```

Examples

Instantiate an instance of the ModelCatalog class

```
>>> model_catalog = ModelCatalog(hbp_username)
```

get_model(*model_id=''*, *alias=''*, *instances=True*, *images=True*)

Retrieve a specific model description by its model_id or alias.

A specific model description can be retrieved from the model catalog in the following ways (in order of priority):

1. specify the *model_id*
2. specify the *alias* (of the model)

Parameters

- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **instances** (*boolean, optional*) – Set to False if you wish to omit the details of the model instances; default True.
- **images** (*boolean, optional*) – Set to False if you wish to omit the details of the model images (figures); default True.

Returns Entire model description as a JSON object.

Return type dict

Examples

```
>>> model = model_catalog.get_model(model_id="8c7cb9f6-e380-452c-9e98-
↪e77254b088c5")
>>> model = model_catalog.get_model(alias="B1")
```

list_models(***filters*)

Retrieve list of model descriptions satisfying specified filters.

The filters may specify one or more attributes that belong to a model description. The following model attributes can be specified:

- app_id
- name
- alias
- author
- organization
- species
- brain_region
- cell_type
- model_scope
- abstraction_level
- owner
- project
- license

Parameters ****filters** (*variable length keyword arguments*) – To be used to filter model descriptions from the model catalog.

Returns List of model descriptions satisfying specified filters.

Return type list

Examples

```
>>> models = model_catalog.list_models()
>>> models = model_catalog.list_models(app_id="39968")
>>> models = model_catalog.list_models(cell_type="Pyramidal Cell", brain_
↪region="Hippocampus")
```

register_model(*app_id=''*, *name=''*, *alias=None*, *author=''*, *organization=''*, *private=False*, *species=''*, *brain_region=''*, *cell_type=''*, *model_scope=''*, *abstraction_level=''*, *owner=''*, *project=''*, *license=''*, *description=''*, *instances=[]*, *images=[]*)

Register a new model in the model catalog.

This allows you to add a new model to the model catalog. Model instances and/or images (figures) can optionally be specified at the time of model creation, or can be added later individually.

Parameters

- **app_id** (*string*) – Specifies the ID of the host model catalog app on the HBP Collaboratory. (the model would belong to this app)

- **name** (*string*) – Name of the model description to be created.
- **alias** (*string, optional*) – User-assigned unique identifier to be associated with model description.
- **author** (*string*) – Name of person creating the model description.
- **organization** (*string, optional*) – Option to tag model with organization info.
- **private** (*boolean*) – Set visibility of model description. If True, model would only be seen in host app (where created). Default False.
- **species** (*string*) – The species for which the model is developed.
- **brain_region** (*string*) – The brain region for which the model is developed.
- **cell_type** (*string*) – The type of cell for which the model is developed.
- **model_scope** (*string*) – Specifies the type of the model.
- **abstraction_level** (*string*) – Specifies the model abstraction level.
- **owner** (*string*) – Specifies the owner of the model. Need not necessarily be the same as the author.
- **project** (*string*) – Can be used to indicate the project to which the model belongs.
- **license** (*string*) – Indicates the license applicable for this model.
- **description** (*string*) – Provides a description of the model.
- **instances** (*list, optional*) – Specify a list of instances (versions) of the model.
- **images** (*list, optional*) – Specify a list of images (figures) to be linked to the model.

Returns UUID of the model description that has been created.

Return type UUID

Examples

(without instances and images)

```
>>> model = model_catalog.register_model(app_id="39968", name="Test Model - B2
↪",
                alias="Model vB2", author="Shailesh Appukuttan", organization=
↪"HBP-SP6",
                private=False, cell_type="Granule Cell", model_scope="Single␣
↪cell model",
                abstraction_level="Spiking neurons",
                brain_region="Basal Ganglia", species="Mouse (Mus musculus)",
                owner="Andrew Davison", project="SP 6.4", license="BSD 3-
↪Clause",
                description="This is a test entry")
```

(with instances and images)

```
>>> model = model_catalog.register_model(app_id="39968", name="Test Model - C2
↪",
                alias="Model vC2", author="Shailesh Appukuttan", organization=
↪"HBP-SP6",
                private=False, cell_type="Granule Cell", model_scope="Single␣
↪cell model",
```

(continues on next page)

(continued from previous page)

```
                abstraction_level="Spiking neurons",
                brain_region="Basal Ganglia", species="Mouse (Mus musculus)",
                owner="Andrew Davison", project="SP 6.4", license="BSD 3-
↪Clause",
                description="This is a test entry! Please ignore.",
                instances=[{"source":"https://www.abcde.com",
                            "version":"1.0", "parameters":""},
                           {"source":"https://www.12345.com",
                            "version":"2.0", "parameters":""}],
                images=[{"url":"http://www.neuron.yale.edu/neuron/sites/
↪default/themes/xchameleon/logo.png",
                         "caption":"NEURON Logo"},
                        {"url":"https://collab.humanbrainproject.eu/assets/
↪hbp_diamond_120.png",
                         "caption":"HBP Logo"}])
```

edit_model(*model_id=”*, *app_id=None*, *name=None*, *alias=None*, *author=None*, *organization=None*, *private=None*, *cell_type=None*, *model_scope=None*, *abstraction_level=None*, *brain_region=None*, *species=None*, *owner=”*, *project=”*, *license=”*, *description=None*)

Edit an existing model on the model catalog.

This allows you to edit a new model to the model catalog. The *model_id* must be provided. Any of the other parameters maybe updated. Only the parameters being updated need to be specified.

Parameters

- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **app_id** (*string*) – Specifies the ID of the host model catalog app on the HBP Collaboratory. (the model would belong to this app)
- **name** (*string*) – Name of the model description to be created.
- **alias** (*string, optional*) – User-assigned unique identifier to be associated with model description.
- **author** (*string*) – Name of person creating the model description.
- **organization** (*string, optional*) – Option to tag model with organization info.
- **private** (*boolean*) – Set visibility of model description. If True, model would only be seen in host app (where created). Default False.
- **species** (*string*) – The species for which the model is developed.
- **brain_region** (*string*) – The brain region for which the model is developed.
- **cell_type** (*string*) – The type of cell for which the model is developed.
- **model_scope** (*string*) – Specifies the type of the model.
- **abstraction_level** (*string*) – Specifies the model abstraction level.
- **owner** (*string*) – Specifies the owner of the model. Need not necessarily be the same as the author.
- **project** (*string*) – Can be used to indicate the project to which the model belongs.
- **license** (*string*) – Indicates the license applicable for this model.
- **description** (*string*) – Provides a description of the model.

Note: Model instances and images (figures) cannot be edited here. This has to be done using *edit_model_instance()* and *edit_model_image()*

Returns UUID of the model description that has been edited.

Return type UUID

Examples

```
>>> model = model_catalog.edit_model(app_id="39968", name="Test Model - B2",
                model_id="8c7cb9f6-e380-452c-9e98-e77254b088c5",
                alias="Model-B2", author="Shailesh Appukuttan", organization=
↪"HBP-SP6",
                private=False, cell_type="Granule Cell", model_scope="Single␣
↪cell model",
                abstraction_level="Spiking neurons",
                brain_region="Basal Ganglia", species="Mouse (Mus musculus)",
                owner="Andrew Davison", project="SP 6.4", license="BSD 3-
↪Clause",
                description="This is a test entry")
```

delete_model (*model_id=''*, *alias=''*)

ONLY FOR SUPERUSERS: Delete a specific model description by its model_id or alias.

A specific model description can be deleted from the model catalog, along with all associated model instances, images and results, in the following ways (in order of priority):

1. specify the *model_id*
2. specify the *alias* (of the model)

Parameters

- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.

Note:

- This feature is only for superusers!

Examples

```
>>> model_catalog.delete_model(model_id="8c7cb9f6-e380-452c-9e98-e77254b088c5
↪")
>>> model_catalog.delete_model(alias="B1")
```

get_attribute_options (*param=''*)

Retrieve valid values for attributes.

Will return the list of valid values (where applicable) for various attributes. The following model attributes can be specified:

- cell_type
- brain_region
- model_scope
- abstraction_level
- species
- organization

If an attribute is specified then, only values that correspond to it will be returned, else values for all attributes are returned.

Parameters **param** (*string, optional*) – Attribute of interest

Returns Dictionary with key(s) as attribute(s), and value(s) as list of valid options.

Return type dict

Examples

```
>>> data = model_catalog.get_attribute_options()
>>> data = model_catalog.get_attribute_options("cell_type")
```

get_model_instance(*instance_path=''*, *instance_id=''*, *model_id=''*, *alias=''*, *version=''*)

Retrieve an existing model instance.

A specific model instance can be retrieved in the following ways (in order of priority):

1. load from a local JSON file specified via *instance_path*
2. specify *instance_id* corresponding to model instance in model catalog
3. specify *model_id* and *version*
4. specify *alias* (of the model) and *version*

Parameters

- **instance_path** (*string*) – Location of local JSON file with model instance metadata.
- **instance_id** (*UUID*) – System generated unique identifier associated with model instance.
- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **version** (*string*) – User-assigned identifier (unique for each model) associated with model instance.

Returns Information about the model instance.

Return type dict

Examples

```
>>> model_instance = model_catalog.get_model_instance(instance_id="a035f2b2-
↪fe2e-42fd-82e2-4173a304263b")
```

(continues on next page)

(continued from previous page)

download_model_instance (*instance_path=''*, *instance_id=''*, *model_id=''*, *alias=''*, *version=''*, *local_directory='.'*)

Download files/directory corresponding to an existing model instance.

Files/directory corresponding to a model instance to be downloaded. The model instance can be specified in the following ways (in order of priority):

1. load from a local JSON file specified via *instance_path*
2. specify *instance_id* corresponding to model instance in model catalog
3. specify *model_id* and *version*
4. specify *alias* (of the model) and *version*

Parameters

- **instance_path** (*string*) – Location of local JSON file with model instance metadata.
- **instance_id** (*UUID*) – System generated unique identifier associated with model instance.
- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **version** (*string*) – User-assigned identifier (unique for each model) associated with model instance.
- **local_directory** (*string*) – Directory path (relative/absolute) where files should be downloaded and saved. Default is current location. Existing files, if any, at the target location will be overwritten!

Returns Absolute path of the downloaded file/directory.

Return type string

Note: Existing files, if any, at the target location will be overwritten!

Examples

```
>>> file_path = model_catalog.download_model_instance(instance_id="a035f2b2-
↪fe2e-42fd-82e2-4173a304263b")
```

list_model_instances (*instance_path=''*, *model_id=''*, *alias=''*)

Retrieve list of model instances belonging to a specified model.

This can be retrieved in the following ways (in order of priority):

1. load from a local JSON file specified via *instance_path*
2. specify *model_id*
3. specify *alias* (of the model)

Parameters

- **instance_path** (*string*) – Location of local JSON file with model instance metadata.
- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.

Returns List of dicts containing information about the model instances.

Return type list

Examples

```
>>> model_instances = model_catalog.list_model_instances(alias="Model vB2")
```

add_model_instance(*model_id=''*, *alias=''*, *source=''*, *version=''*, *description=''*, *parameters=''*, *code_format=''*, *hash=''*, *morphology=''*)

Register a new model instance.

This allows to add a new instance of an existing model in the model catalog. The *model_id* needs to be specified as input parameter.

Parameters

- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **source** (*string*) – Path to model source code repository (e.g. github).
- **version** (*string*) – User-assigned identifier (unique for each model) associated with model instance.
- **description** (*string, optional*) – Text describing this specific model instance.
- **parameters** (*string, optional*) – Any additional parameters to be submitted to model, or used by it, at runtime.
- **code_format** (*string, optional*) – Indicates the language/platform in which the model was developed.
- **hash** (*string, optional*) – Similar to a checksum; can be used to identify model instances from their implementation.
- **morphology** (*string / list, optional*) – URL(s) to the morphology file(s) employed in this model.

Returns UUID of the model instance that has been created.

Return type UUID

Note:

- *alias* is not currently implemented in the API; kept for future use.
- TODO: Either model_id or alias needs to be provided, with model_id taking precedence over alias.

Examples

```
>>> instance_id = model_catalog.add_model_instance(model_id="196b89a3-e672-
↪4b96-8739-748ba3850254",
                                                   source="https://www.abcde.com",
                                                   version="1.0",
                                                   description="basic model variant",
                                                   parameters="",
                                                   code_format="py",
                                                   hash="",
                                                   morphology="")
```

find_model_instance_else_add(*model_obj*)

Find existing model instance; else create a new instance

This checks if the input model object has an associated model instance. If not, a new model instance is created.

Parameters **model_obj** (*object*) – Python object representing a model.

Returns UUID of the existing or created model instance.

Return type UUID

Note:

- *model_obj* is expected to contain the attribute *model_instance_uuid*, or both the attributes *model_uuid* and *model_version*.

Examples

```
>>> instance_id = model_catalog.find_model_instance_else_add(model)
```

edit_model_instance(*instance_id=''*, *model_id=''*, *alias=''*, *source=None*, *version=None*, *description=None*, *parameters=None*, *code_format=None*, *hash=None*, *morphology=None*)

Edit an existing model instance.

This allows to edit an instance of an existing model in the model catalog. The model instance can be specified in the following ways (in order of priority):

1. specify *instance_id* corresponding to model instance in model catalog
2. specify *model_id* and *version*
3. specify *alias* (of the model) and *version*

Only the parameters being updated need to be specified. You cannot edit the model *version* in the latter two cases. To do so, you must employ the first option above. You can retrieve the *instance_id* via *get_model_instance()*

Parameters

- **instance_id** (*UUID*) – System generated unique identifier associated with model instance.
- **model_id** (*UUID*) – System generated unique identifier associated with model description.

- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **source** (*string*) – Path to model source code repository (e.g. github).
- **version** (*string*) – User-assigned identifier (unique for each model) associated with model instance.
- **description** (*string, optional*) – Text describing this specific model instance.
- **parameters** (*string, optional*) – Any additional parameters to be submitted to model, or used by it, at runtime.
- **code_format** (*string, optional*) – Indicates the language/platform in which the model was developed.
- **hash** (*string, optional*) – Similar to a checksum; can be used to identify model instances from their implementation.
- **morphology** (*string / list, optional*) – URL(s) to the morphology file(s) employed in this model.

Returns UUID of the model instance that has been edited.

Return type UUID

Examples

```
>>> instance_id = model_catalog.edit_model_instance(instance_id="fd1ab546-
↪80f7-4912-9434-3c62af87bc77",
                                                    source="https://www.abcde.com",
                                                    version="1.0",
                                                    description="passive model variant",
                                                    parameters="",
                                                    code_format="py",
                                                    hash="",
                                                    morphology="")
```

delete_model_instance(*instance_id=''*, *model_id=''*, *alias=''*, *version=''*)

ONLY FOR SUPERUSERS: Delete an existing model instance.

This allows to delete an instance of an existing model in the model catalog. The model instance can be specified in the following ways (in order of priority):

1. specify *instance_id* corresponding to model instance in model catalog
2. specify *model_id* and *version*
3. specify *alias* (of the model) and *version*

Parameters

- **instance_id** (*UUID*) – System generated unique identifier associated with model instance.
- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **version** (*string*) – User-assigned unique identifier associated with model instance.

Note:

- This feature is only for superusers!

Examples

```
>>> model_catalog.delete_model_instance(model_id="8c7cb9f6-e380-452c-9e98-
↪e77254b088c5")
>>> model_catalog.delete_model_instance(alias="B1", version="1.0")
```

get_model_image(*image_id=''*)

Retrieve image info from a model description.

This allows to retrieve image (figure) info from the model catalog. The *image_id* needs to be specified as input parameter.

Parameters **image_id** (*UUID*) – System generated unique identifier associated with image (figure).

Returns Information about the image (figure) retrieved.

Return type dict

Examples

```
>>> model_image = model_catalog.get_model_image(image_id="2b45e7d4-a7a1-4a31-
↪a287-aee7072e3e75")
```

list_model_images(*model_id=''*, *alias=''*)

Retrieve all images (figures) associated with a model.

This can be retrieved in the following ways (in order of priority):

1. specify *model_id*
2. specify *alias* (of the model)

Parameters

- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.

Returns List of dicts containing information about the model images (figures).

Return type list

Examples

```
>>> model_images = model_catalog.list_model_images(model_id="196b89a3-e672-
↪4b96-8739-748ba3850254")
```

add_model_image(*model_id=''*, *alias=''*, *url=''*, *caption=''*)
Add a new image (figure) to a model description.

This allows to add a new image (figure) to an existing model in the model catalog. The *model_id* needs to be specified as input parameter.

Parameters

- **model_id** (*UUID*) – System generated unique identifier associated with model description.
- **alias** (*string*) – User-assigned unique identifier associated with model description.
- **url** (*string*) – Url of image (figure) to be added.
- **caption** (*string*) – Caption to be associated with the image (figure).

Returns UUID of the image (figure) that was added.

Return type UUID

Note:

- *alias* is not currently implemented in the API; kept for future use.
- TODO: Either model_id or alias needs to be provided, with model_id taking precedence over alias.
- TODO: Allow image (figure) to be located locally

Examples

```
>>> image_id = model_catalog.add_model_image(model_id="196b89a3-e672-4b96-
↪8739-748ba3850254",
                                             url="http://www.neuron.yale.edu/neuron/
↪sites/default/themes/xchameleon/logo.png",
                                             caption="NEURON Logo")
```

edit_model_image(*image_id=''*, *url=None*, *caption=None*)
Edit an existing image (figure) metadata.

This allows to edit the metadata of an image (figure) in the model catalog. The *image_id* needs to be specified as input parameter. Only the parameters being updated need to be specified.

Parameters **image_id** (*UUID*) – System generated unique identifier associated with image (figure).

Returns UUID of the image (figure) that was edited.

Return type UUID

Examples

```
>>> image_id = model_catalog.edit_model_image(image_id="2b45e7d4-a7a1-4a31-
↪a287-aee7072e3e75", caption = "Some Logo", url="http://www.somesite.com/
↪logo.png")
```

delete_model_image(*image_id=''*)
ONLY FOR SUPERUSERS: Delete an image from a model description.

This allows to delete an image (figure) info from the model catalog. The *image_id* needs to be specified as input parameter.

Parameters **image_id** (*UUID*) – System generated unique identifier associated with image (figure).

Note:

- This feature is only for superusers!

Examples

```
>>> model_catalog.delete_model_image(image_id="2b45e7d4-a7a1-4a31-a287-
↪aee7072e3e75")
```

CHAPTER 6

Utilities

Miscellaneous methods that help in different aspects of model validation. Does not require explicit instantiation.

The following methods are available:

Action	Method
View JSON data in web browser	*view_json_tree()*
Prepare test for execution	*prepare_run_test_offline()*
Run the validation test	*run_test_offline()*
Register result with validation service	*upload_test_result()*
Run test and register result	*run_test()*
Download PDF report of test results	*generate_report()*
Obtain score matrix for test results	*generate_score_matrix()*
Get Pandas DataFrame from score matrix	*get_raw_dataframe()*
Display score matrix in web browser	*display_score_matrix_html()*

hbp_validation_framework.utils.**view_json_tree**(*data*)

Displays the JSON tree structure inside the web browser

This method can be used to view any JSON data, generated by any of the validation client's methods, in a tree-like representation.

Parameters **data** (*string*) – JSON object represented as a string.

Returns Does not return any data. JSON displayed inside web browser.

Return type None

Examples

```
>>> model = model_catalog.get_model(alias="HCkt")
>>> from hbp_validation_framework import utils
>>> utils.view_json_tree(model)
```

hbp_validation_framework.utils.**prepare_run_test_offline**(*username=''*, *password=None*, *environment='production'*, *test_instance_id=''*, *test_id=''*, *test_alias=''*, *test_version=''*, *client_obj=None*, ***params*)

Gather info necessary for running validation test

This method will select the specified test and prepare a config file enabling offline execution of the validation test. The observation file required by the test is also downloaded and stored locally. The test can be specified in the following ways (in order of priority):

1. specify *test_instance_id* corresponding to test instance in test library
2. specify *test_id* and *test_version*
3. specify *test_alias* and *test_version* Note: for (2) and (3) above, if *test_version* is not specified,

   then the latest test version is retrieved

Parameters

- **username** (*string*) – Your HBP Collaboratory username.
- **password** (*string*) – Your HBP Collaboratory password.
- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system, which is appropriate for most users. When set to *dev*, it uses the *development* system. For other values, an external config file would be read (the latter is currently not implemented).
- **test_instance_id** (*UUID*) – System generated unique identifier associated with test instance.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **test_alias** (*string*) – User-assigned unique identifier associated with test definition.
- **test_version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.
- **client_obj** (*ModelCatalog/TestLibrary object*) – Used to easily create a new ModelCatalog/TestLibrary object if either exist already. Avoids need for repeated authentications; improves performance. Also, helps minimize being blocked out by the authentication server for repeated authentication requests (applicable when running several tests in quick succession, e.g. in a loop).
- ****params** (*list*) – Keyword arguments to be passed to the Test constructor.

Note: Should be run on node having access to external URLs (i.e. with internet access)

Returns The absolute path of the generated test config file

Return type path

Examples

```
>>> test_config_file = utils.prepare_run_test_offline(username="shailesh", test_
↪alias="CDT-5", test_version="5.0")
```

hbp_validation_framework.utils.**run_test_offline**(*model=''*, *test_config_file=''*)

Run the validation test

This method will accept a model, located locally, run the test specified via the test config file (generated by *prepare_run_test_offline()*), and store the results locally.

Parameters

- **model** (*sciunit.Model*) – A `sciunit.Model` instance.
- **test_config_file** (*string*) – Absolute path of the test config file generated by *prepare_run_test_offline()*

Note: Can be run on node(s) having no access to external URLs (i.e. without internet access). Also, it is required that the test_config_file and the test_observation_file are located in the same directory.

Returns The absolute path of the generated test result file

Return type path

Examples

```
>>> test_result_file = utils.run_test_offline(model=model, test_config_file=test_
↪config_file)
```

hbp_validation_framework.utils.**upload_test_result**(*username=''*, *password=None*, *environment='production'*, *test_result_file=''*, *storage_collab_id=''*, *register_result=True*, *client_obj=None*)

Register the result with the Validation Service

This method will register the validation result specified via the test result file (generated by *run_test_offline()*) with the validation service.

Parameters

- **username** (*string*) – Your HBP Collaboratory username.
- **password** (*string*) – Your HBP Collaboratory password.
- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system, which is appropriate for most users. When set to *dev*, it uses the *development* system. For other values, an external config file would be read (the latter is currently not implemented).
- **test_result_file** (*string*) – Absolute path of the test result file generated by *run_test_offline()*
- **storage_collab_id** (*string*) – Collab ID where output files should be stored; if empty, stored in model's host Collab.

- **register_result** (*boolean*) – Specify whether the test results are to be scored on the validation framework. Default is set as True.
- **client_obj** (*ModelCatalog/TestLibrary object*) – Used to easily create a new ModelCatalog/TestLibrary object if either exist already. Avoids need for repeated authentications; improves performance. Also, helps minimize being blocked out by the authentication server for repeated authentication requests (applicable when running several tests in quick succession, e.g. in a loop).

Note: Should be run on node having access to external URLs (i.e. with internet access)

Returns

- *UUID* – UUID of the test result that has been created.
- *object* – score object evaluated by the test.

Examples

```
>>> result_id, score = utils.upload_test_result(username="shailesh", test_result_
↪file=test_result_file)
```

hbp_validation_framework.utils.**run_test**(*username=''*, *password=None*, *environment='production'*, *model=''*, *test_instance_id=''*, *test_id=''*, *test_alias=''*, *test_version=''*, *storage_collab_id=''*, *register_result=True*, *client_obj=None*, ***params*)

Run validation test and register result

This will execute the following methods by relaying the output of one to the next: 1. *prepare_run_test_offline()* 2. *run_test_offline()* 3. *upload_test_result()*

Parameters

- **username** (*string*) – Your HBP Collaboratory username.
- **password** (*string*) – Your HBP Collaboratory password.
- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system, which is appropriate for most users. When set to *dev*, it uses the *development* system. For other values, an external config file would be read (the latter is currently not implemented).
- **model** (*sciunit.Model*) – A `sciunit.Model` instance.
- **test_instance_id** (*UUID*) – System generated unique identifier associated with test instance.
- **test_id** (*UUID*) – System generated unique identifier associated with test definition.
- **test_alias** (*string*) – User-assigned unique identifier associated with test definition.
- **test_version** (*string*) – User-assigned identifier (unique for each test) associated with test instance.
- **storage_collab_id** (*string*) – Collab ID where output files should be stored; if empty, stored in model's host Collab.

- **register_result** (*boolean*) – Specify whether the test results are to be scored on the validation framework. Default is set as True.
- **client_obj** (*ModelCatalog/TestLibrary object*) – Used to easily create a new ModelCatalog/TestLibrary object if either exist already. Avoids need for repeated authentications; improves performance. Also, helps minimize being blocked out by the authentication server for repeated authentication requests (applicable when running several tests in quick succession, e.g. in a loop).
- ****params** (*list*) – Keyword arguments to be passed to the Test constructor.

Note: Should be run on node having access to external URLs (i.e. with internet access)

Returns

- *UUID* – UUID of the test result that has been created.
- *object* – score object evaluated by the test.

Examples

```
>>> result_id, score = utils.run_test(username="HBP_USERNAME", password="HBP_
↪PASSWORD" environment="production", model=cell_model, test_alias="basalg_msn_d1
↪", test_version="1.0", storage_collab_id="8123", register_result=True)
```

hbp_validation_framework.utils.**generate_report**(*username=''*, *password=None*, *environment='production'*, *result_list=[]*, *only_combined=True*, *client_obj=None*)

Generates and downloads a PDF report of test results

This method will generate and download a PDF report of the specified test results. The report will consist of all information relevant to that particular result, such as:

- result info
- model info
- model instance info
- test info
- test instance info
- output files associated with result

Parameters

- **username** (*string*) – Your HBP collaboratory username.
- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system, which is appropriate for most users. When set to *dev*, it uses the *development* system. For other values, an external config file would be read (the latter is currently not implemented).
- **result_list** (*list*) – List of result UUIDs that need to be included in report.

- **only_combined** (*boolean, optional*) – Indicates whether only a single combined PDF should be saved. Set to *True* as default. When set to *False*, then *n+2* PDFs will be saved, where *n* is the number of valid result UUIDs. These would include:
  - Combined PDF report
  - Summary of call to *generate_report()*
  - One PDF for each valid result UUID
- **client_obj** (*ModelCatalog/TestLibrary object*) – Used to easily create a new ModelCatalog/TestLibrary object if either exist already. Avoids need for repeated authentications; improves performance. Also, helps minimize being blocked out by the authentication server for repeated authentication requests (applicable when running several tests in quick succession, e.g. in a loop).

Returns

- *list* – List of valid UUIDs for which the PDF report was generated
- *path* – The absolute path of the generated report

Examples

```
>>> result_list = ["a618a6b1-e92e-4ac6-955a-7b8c6859285a", "793e5852-761b-4801-
↪84cb-53af6f6c1acf"]
>>> valid_uuids, report_path = utils.generate_report(username="shailesh", result_
↪list=result_list)
```

hbp_validation_framework.utils.**generate_score_matrix**(*username=''*, *password=None*, *environment='production'*, *model_list=[]*, *model_instance_list=[]*, *test_list=[]*, *test_instance_list=[]*, *result_list=[]*, *collab_id=None*, *client_obj=None*)

Generates a styled pandas dataframe with score matrix

This method will generate a styled pandas dataframe for the specified test results. Each row will correspond to a particular model instance, and the columns correspond to the test instances.

Parameters

- **username** (*string*) – Your HBP collaboratory username.
- **environment** (*string, optional*) – Used to indicate whether being used for development/testing purposes. Set as *production* as default for using the production system, which is appropriate for most users. When set to *dev*, it uses the *development* system. For other values, an external config file would be read (the latter is currently not implemented).
- **model_list** (*list*) – List of model UUIDs or aliases for which score matrix is to be generated.
- **model_instance_list** (*list*) – List of model instance UUIDs for which score matrix is to be generated.
- **test_list** (*list*) – List of test UUIDs or aliases for which score matrix is to be generated.

- **test_instance_list** (*list*) – List of test instance UUIDs for which score matrix is to be generated.
- **result_list** (*list*) – List of result UUIDs for which score matrix is to be generated.
- **collab_id** (*string, optional*) – Collaboratory ID where hyperlinks to results are to be redirected. If unspecified, the scores will not have clickable hyperlinks.
- **client_obj** (*ModelCatalog/TestLibrary object*) – Used to easily create a new ModelCatalog/TestLibrary object if either exist already. Avoids need for repeated authentications; improves performance. Also, helps minimize being blocked out by the authentication server for repeated authentication requests (applicable when running several tests in quick succession, e.g. in a loop).

Note: Only the latest score entry from specified input for a particular model instance and test instance combination will be selectedself. To get the raw (unstyled) dataframe, use *get_raw_dataframe()*

Returns

- *pandas.io.formats.style.Styler* – A 2-dimensional matrix representation of the scores
- *list* – List of entries from specified input that could not be resolved and thus ignored

Examples

```
>>> result_list = ["a618a6b1-e92e-4ac6-955a-7b8c6859285a", "793e5852-761b-4801-
↪84cb-53af6f6c1acf"]
>>> styled_df, excluded = utils.generate_score_matrix(username="shailesh", result_
↪list=result_list)
```

hbp_validation_framework.utils.**get_raw_dataframe**(*styled_df*)

Creates DataFrame from output of :meth‘generate_score_matrix‘

This method creates a raw DataFrame objects from its styled variant as generated by :meth‘generate_score_matrix‘. The cell values in latter could contain additional data (i.e. result UUIDs) for creating hyperlinks. This is filtered out here such that the cell values only contain scores.

Parameters **styled_df** (*pandas.io.formats.style.Styler*) – Styled DataFrame object generated by :meth‘generate_score_matrix‘

Returns A 2-dimensional matrix representation of the scores without any formatting

Return type pandas.core.frame.DataFrame

Examples

```
>>> df = utils.get_raw_dataframe(styled_df)
```

hbp_validation_framework.utils.**display_score_matrix_html**(*styled_df=None*, *df=None*)

Displays score matrix generated from :meth‘generate_score_matrix‘ inside web browser

This method displays the scoring matrix generated by :meth‘generate_score_matrix‘ inside a web browser. Input can either be the styled DataFrame object generated by :meth‘generate_score_matrix‘ or the raw DataFrame object from :meth‘get_raw_dataframe‘.

Parameters

- **styled_df** (*pandas.io.formats.style.Styler*) – Styled DataFrame object generated by :meth‘generate_score_matrix‘
- **df** (*pandas.core.frame.DataFrame*) – DataFrame object generated by :meth‘get_raw_dataframe‘

Returns Does not return any data. JSON displayed inside web browser.

Return type None

Examples

```
>>> utils.display_score_matrix_html(styled_df)
```

Python Module Index

h

Index