
microdrop Documentation

Release 2.0.post81.dev72981766

Christian Fobel

September 09, 2016

1 Project Modules	3
1.1 microdrop Package	3
2 Indices and tables	13
Python Module Index	15

Contents:

Project Modules

1.1 microdrop Package

1.1.1 microdrop Package

`microdrop.__init__.base_path()`

`microdrop.__init__.glade_path()`

Return path to *.glade* files used by *gtk* to construct views.

1.1.2 `__main__` Module

1.1.3 `app` Module

1.1.4 `app_context` Module

Copyright 2011 Ryan Fobel

This file is part of Microdrop.

Microdrop is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by Foundation, either version 3 of the License, or (at your option) any later version.

Microdrop is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Microdrop. If not, see <http://www.gnu.org/licenses/>.

`microdrop.app_context.get_app()`

`microdrop.app_context.get_hub_uri()`

1.1.5 `config` Module

Copyright 2011 Ryan Fobel

This file is part of Microdrop.

Microdrop is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Microdrop is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Microdrop. If not, see <http://www.gnu.org/licenses/>.

```
class microdrop.config.Config (filename=None)
    Bases: object
```

Methods

```
default_config_directory = path('/home/docs/.microdrop')
```

```
default_config_path = path('/home/docs/.microdrop/microdrop.ini')
```

```
load (filename=None)
```

Load a Config object from a file.

Parameters filename – path to file. If None, try loading from the default location, and if there's no file, create a Config object with the default options.

Raises

- `IOError` – The file does not exist.
- `ConfigObjError` – There was a problem parsing the config file.
- `ValidationError` – There was a problem validating one or more fields.

```
save (filename=None)
```

```
spec = '\n [dmf_device]\n # name of the most recently used DMF device\n name = string(default=None)\n\n [protocol]\n #
```

```
exception microdrop.config.ValidationError
    Bases: exceptions.Exception
```

1.1.6 dmf_device Module

1.1.7 experiment_log Module

Copyright 2011 Ryan Fobel

This file is part of Microdrop.

Microdrop is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

Microdrop is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with Microdrop. If not, see <http://www.gnu.org/licenses/>.

```
class microdrop.experiment_log.ExperimentLog (directory=None)
```


Methods

add_data (*data*, *plugin_name*='core')

add_step (*step_number*, *attempt*=0)

class_version = '0.3.0'

get (*name*, *plugin_name*='core')

get_log_path ()

classmethod load (*filename*)

Load an experiment log from a file.

Parameters **filename** – path to file.

Raises

- `TypeError` – file is not an experiment log.
- `FutureVersionError` – file was written by a future version of the software.

save (*filename*=None, *format*='pickle')

start_time ()

1.1.8 interfaces Module

class `microdrop.interfaces.IFoo`

Bases: `pyutilib.component.core.core.Interface`

class `microdrop.interfaces.ILoggingPlugin`

Bases: `pyutilib.component.core.core.Interface`

Methods

on_critical (*record*)

on_debug (*record*)

on_error (*record*)

on_info (*record*)

on_warning (*record*)

class `microdrop.interfaces.IPlugin`

Bases: `pyutilib.component.core.core.Interface`

Methods

get_schedule_requests (*function_name*)

Parameters **function_name** (*str*) – Plugin callback function name.

Returns List of scheduling requests (i.e., `ScheduleRequest` instances) for the function specified by `function_name`.

Return type `list`

`get_step_form_class()`

`get_step_values(step_number=None)`

`on_app_exit()`

Handler called just before the Microdrop application exits.

`on_app_options_changed(plugin_name)`

Handler called when the app options are changed for a particular plugin. This will, for example, allow for GUI elements to be updated.

Parameters `plugin` (*str*) – Plugin name for which the app options changed

`on_dmf_device_changed(dmf_device)`

Handler called when a DMF device is modified (e.g., channel assignment, scaling, etc.).

Parameters `dmf_device` (*microdrop.dmf_device.DmfDevice*) –

`on_dmf_device_saved(dmf_device)`

Handler called when a DMF device is saved.

Parameters `dmf_device` (*microdrop.dmf_device.DmfDevice*) –

`on_dmf_device_swapped(old_dmf_device, dmf_device)`

Handler called when a different DMF device is swapped in (e.g., when a new device is loaded).

Parameters

- `old_dmf_device` (*microdrop.dmf_device.DmfDevice*) – Original device.
- `dmf_device` (*microdrop.dmf_device.DmfDevice*) – New device.

`on_experiment_log_changed(experiment_log)`

Handler called when the current experiment log changes (e.g., when a protocol finishes running).

Parameters `experiment_log` (*microdrop.experiment_log.ExperimentLog*) – Reference to new experiment log instance.

`on_experiment_log_selection_changed(data)`

Handler called whenever the experiment log selection changes.

Parameters `data` – experiment log data (list of dictionaries, one per step) for the selected steps

`on_export_experiment_log_data(experiment_log)`

Handler called when the experiment log is exported.

Parameters `log` – experiment log data (list of dictionaries, one per step) for the selected steps

Returns A dictionary of pandas.DataFrame objects containing any relevant data that should be exported by the plugin, each keyed by a unique name.

`on_metadata_changed(schema, original_metadata, metadata)`

Handler called each time the experiment metadata has changed.

Parameters

- `schema` (*dict*) – jsonschema schema definition for metadata.
- `original_metadata` – Original metadata.
- `metadata` – New metadata matching schema

`on_plugin_disable()`

Handler called once the plugin instance is disabled.

on_plugin_disabled (*env, plugin*)

Handler called to notify that a plugin has been disabled.

Note that this signal is broadcast to all plugins implementing the *IPlugin* interface, whereas the *on_plugin_disable()* method is called directly on the plugin that is being disabled.

Parameters

- **env** (*str*) – pyutilib plugin environment.
- **plugin** (*str*) – Plugin name.

on_plugin_enable ()

Handler called once the plugin instance is enabled.

Note: if you inherit your plugin from *AppDataController* and don't implement this handler, by default, it will automatically load all app options from the config file. If you decide to override the default handler, you should call:

```
AppDataController.on_plugin_enable(self)
```

to retain this functionality.

on_plugin_enabled (*env, plugin*)

Handler called to notify that a plugin has been enabled.

Note that this signal is broadcast to all plugins implementing the *IPlugin* interface, whereas the *on_plugin_enable()* method is called directly on the plugin that is being enabled.

Parameters

- **env** (*str*) – pyutilib plugin environment.
- **plugin** (*str*) – Plugin name.

on_protocol_changed ()

Handler called when a protocol is modified.

on_protocol_pause ()

Handler called when a protocol is paused.

on_protocol_run ()

Handler called when a protocol starts running.

on_protocol_swapped (*old_protocol, protocol*)

Handler called when a different protocol is swapped in (e.g., when a protocol is loaded or a new protocol is created).

Parameters

- **old_protocol** (*microdrop.protocol.Protocol*) – Original protocol.
- **protocol** (*microdrop.protocol.Protocol*) – New protocol.

on_step_complete (*plugin_name, return_value=None*)

Handler called whenever a plugin completes a step.

Returns

- 'Repeat': repeat the step
- 'Fail': unrecoverable error (stop the protocol)

Return type str or None

on_step_created (*step_number*)

Handler called whenever a new step is created.

Parameters `step_number` (*int*) – New step number.

on_step_options_changed (*plugin, step_number*)

Handler called when the step options are changed for a particular plugin. This will, for example, allow for GUI elements to be updated based on step specified.

Parameters

- **plugin** (*SingletonPlugin*) – Plugin instance for which the step options changed.
- **step_number** (*int*) – Step number that the options changed for.

on_step_options_swapped (*plugin, old_step_number, step_number*)

Handler called when the step options are changed for a particular plugin. This will, for example, allow for GUI elements to be updated based on step specified.

Parameters

- **plugin** (*SingletonPlugin*) – Plugin instance for which the step options changed.
- **old_step_number** (*int*) – Original step number.
- **step_number** (*int*) – New step number.

on_step_run ()

Handler called whenever a step is executed. Note that this signal is only emitted in realtime mode or if a protocol is running.

Plugins that handle this signal must emit the `on_step_complete()` signal once they have completed the step. The protocol controller will wait until all plugins have completed the current step before proceeding.

Returns

- 'Repeat': repeat the step
- 'Fail': unrecoverable error (stop the protocol)

Return type str or None

on_step_swapped (*old_step_number, step_number*)

Handler called when the current step is swapped.

Parameters

- **old_step_number** (*int*) – Original step number.
- **step_number** (*int*) – New step number.

class `microdrop.interfaces.IWaveformGenerator`

Bases: `pyutilib.component.core.core.Interface`

Methods

set_frequency (*frequency*)

Set the waveform frequency.

Parameters `frequency` – frequency in Hz

set_voltage (*voltage*)

Set the waveform voltage.

Parameters `voltage` – RMS voltage

1.1.9 logger Module

1.1.10 microdrop Module

1.1.11 plugin_helpers Module

1.1.12 plugin_manager Module

1.1.13 protocol Module

1.1.14 Subpackages

bin Package

create_portable_config Module

microdrop.bin.create_portable_config.**main** (*output_dir*)

microdrop.bin.create_portable_config.**parse_args** (*args=None*)
Parses arguments, returns (options, args).

latest_versions Module

microdrop.bin.latest_versions.**get_latest_version_content** (*server_url='http://microfluidics.utoronto.ca/updates'*)

core_plugins Package

Subpackages

device_info_plugin Package

device_info_plugin Package

on_plugin_install Module

release Module

rename Module

electrode_controller_plugin Package

electrode_controller_plugin Package

on_plugin_install Module

release Module

rename Module

zmq_hub_plugin Package

zmq_hub_plugin Package

on_plugin_install Module

release Module

rename Module

gui Package

`app_options_controller` Module

`cairo_view` Module

`channel_sweep` Module

`config_controller` Module

`dmf_device_controller` Module

`dmf_device_controller.video` Module

`dmf_device_view.video` Module

`experiment_log_controller` Module

`field_filter_controller` Module

`main_window_controller` Module

`plugin_download_dialog` Module

`plugin_manager_controller` Module

`plugin_manager_dialog` Module

`protocol_controller` Module

`protocol_grid_controller` Module

tests Package

`test_dmf_device` Module

`test_experiment_log` Module

`test_protocol` Module

`update_dmf_control_board` Module

Indices and tables

- `genindex`
- `modindex`
- `search`

m

microdrop.__init__, 3
microdrop.app_context, 3
microdrop.bin.create_portable_config, 9
microdrop.bin.latest_versions, 9
microdrop.config, 3
microdrop.experiment_log, 4
microdrop.interfaces, 5
microdrop.tests.update_dmf_control_board,

11

A

add_data() (microdrop.experiment_log.ExperimentLog method), 5
 add_step() (microdrop.experiment_log.ExperimentLog method), 5

B

base_path() (in module microdrop.__init__), 3

C

class_version (microdrop.experiment_log.ExperimentLog attribute), 5
 Config (class in microdrop.config), 4

D

default_config_directory (microdrop.config.Config attribute), 4
 default_config_path (microdrop.config.Config attribute), 4

E

ExperimentLog (class in microdrop.experiment_log), 4

G

get() (microdrop.experiment_log.ExperimentLog method), 5
 get_app() (in module microdrop.app_context), 3
 get_hub_uri() (in module microdrop.app_context), 3
 get_latest_version_content() (in module microdrop.bin.latest_versions), 9
 get_log_path() (microdrop.experiment_log.ExperimentLog method), 5
 get_schedule_requests() (microdrop.interfaces.IPlugin method), 5
 get_step_form_class() (microdrop.interfaces.IPlugin method), 5
 get_step_values() (microdrop.interfaces.IPlugin method), 6
 glade_path() (in module microdrop.__init__), 3

I

IFoo (class in microdrop.interfaces), 5
 ILoggingPlugin (class in microdrop.interfaces), 5
 IPlugin (class in microdrop.interfaces), 5
 IWaveformGenerator (class in microdrop.interfaces), 8

L

load() (microdrop.config.Config method), 4
 load() (microdrop.experiment_log.ExperimentLog class method), 5

M

main() (in module microdrop.bin.create_portable_config), 9
 microdrop.__init__ (module), 3
 microdrop.app_context (module), 3
 microdrop.bin.create_portable_config (module), 9
 microdrop.bin.latest_versions (module), 9
 microdrop.config (module), 3
 microdrop.experiment_log (module), 4
 microdrop.interfaces (module), 5
 microdrop.tests.update_dmf_control_board (module), 11

O

on_app_exit() (microdrop.interfaces.IPlugin method), 6
 on_app_options_changed() (microdrop.interfaces.IPlugin method), 6
 on_critical() (microdrop.interfaces.ILoggingPlugin method), 5
 on_debug() (microdrop.interfaces.ILoggingPlugin method), 5
 on_dmf_device_changed() (microdrop.interfaces.IPlugin method), 6
 on_dmf_device_saved() (microdrop.interfaces.IPlugin method), 6
 on_dmf_device_swapped() (microdrop.interfaces.IPlugin method), 6
 on_error() (microdrop.interfaces.ILoggingPlugin method), 5

on_experiment_log_changed() (microdrop.interfaces.IPlugin method), 6
 on_experiment_log_selection_changed() (microdrop.interfaces.IPlugin method), 6
 on_export_experiment_log_data() (microdrop.interfaces.IPlugin method), 6
 on_info() (microdrop.interfaces.ILoggingPlugin method), 5
 on_metadata_changed() (microdrop.interfaces.IPlugin method), 6
 on_plugin_disable() (microdrop.interfaces.IPlugin method), 6
 on_plugin_disabled() (microdrop.interfaces.IPlugin method), 6
 on_plugin_enable() (microdrop.interfaces.IPlugin method), 7
 on_plugin_enabled() (microdrop.interfaces.IPlugin method), 7
 on_protocol_changed() (microdrop.interfaces.IPlugin method), 7
 on_protocol_pause() (microdrop.interfaces.IPlugin method), 7
 on_protocol_run() (microdrop.interfaces.IPlugin method), 7
 on_protocol_swapped() (microdrop.interfaces.IPlugin method), 7
 on_step_complete() (microdrop.interfaces.IPlugin method), 7
 on_step_created() (microdrop.interfaces.IPlugin method), 7
 on_step_options_changed() (microdrop.interfaces.IPlugin method), 8
 on_step_options_swapped() (microdrop.interfaces.IPlugin method), 8
 on_step_run() (microdrop.interfaces.IPlugin method), 8
 on_step_swapped() (microdrop.interfaces.IPlugin method), 8
 on_warning() (microdrop.interfaces.ILoggingPlugin method), 5

V

ValidationError, 4

P

parse_args() (in module microdrop.bin.create_portable_config), 9

S

save() (microdrop.config.Config method), 4
 save() (microdrop.experiment_log.ExperimentLog method), 5
 set_frequency() (microdrop.interfaces.IWaveformGenerator method), 8
 set_voltage() (microdrop.interfaces.IWaveformGenerator method), 8
 spec (microdrop.config.Config attribute), 4