microdrop Documentation

Release 2.0.post81.dev72981766

Christian Fobel

September 09, 2016

Contents:

CHAPTER 1

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

`microdrop.app_context.`**`get_app`**`()`

`microdrop.app_context.`**`get_hub_uri`**`()`

1.1.5 `config` Module

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

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 overide 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/upda*

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

CHAPTER 2

Indices and tables

- genindex
- modindex
- search

Index