

msaSDK Module

`.utils.scheduler`

Attributes

`__version__` module-attribute

```
__version__ = '0.1.0'
```

Classes

MSAScheduler

Attributes

`debug` instance-attribute

```
debug = debug
```

Debug mode True/False

`enabled` instance-attribute

```
enabled: bool = False
```

`is_running` instance-attribute

```
is_running: bool = False
```

`jobs` instance-attribute

```
jobs = jobs
```

dictionary MSATimers instances

local_time_zone instance-attribute

```
local_time_zone = local_time_zone
```

logger instance-attribute

```
logger = parent_logger if parent_logger else logger
```

Functions

__init__

```
__init__(
    jobs: dict,
    local_time_zone: str = "UTC",
    poll_millis: float = 1000,
    debug: bool = False,
    parent_logger: Logger = None,
)
```

MSAScheduler object runs timers

Standard Polling is 1 second

PARAMETER	DESCRIPTION
<small>jobs</small>	timer_jobs: dict[MSATimerEnum, list] = {... TYPE: <small>dict</small>
<small>local_time_zone</small>	str = 'UTC' TYPE: <small>str</small> DEFAULT: <small>'UTC'</small>
<small>poll_millis</small>	float = 1000 TYPE: <small>float</small> DEFAULT: <small>1000</small>
<small>debug</small>	bool = False TYPE: <small>bool</small> DEFAULT: <small>False</small>
<small>parent_logger</small>	logger instance to use, if empty it creates a local loguru logger TYPE: <small>Logger</small> DEFAULT: <small>None</small>

run_timers async

```
run_timers(poll_adjuster = 0.99, debug = False)
```

runs timers as follows:

- Step 1: run every poll jobs
- Step 2: load timer queues for next poll
- Step 3: delay function which runs previous poll queues

poll_adjustor allows time for other timing

stop_timers async

```
stop_timers()
```

Stop all timers

MSATimerEnum

Bases: `str`, `Enum`

Enum for the different timer Types

Attributes

every_hour class-attribute

```
every_hour = 'every hour'
```

every_minute class-attribute

```
every_minute = 'every minute'
```

every_poll class-attribute

```
every_poll = 'every poll'
```

every_second class-attribute

```
every_second = 'every second'
```

on_the_15_minute class-attribute

```
on_the_15_minute = 'on the 15 minute'
```

on_the_15_second class-attribute

```
on_the_15_second = 'on the 15 second'
```

on_the_30_minute class-attribute

```
on_the_30_minute = 'on the 30 minute'
```

on_the_30_second class-attribute

```
on_the_30_second = 'on the 30 second'
```

on_the_5_minute class-attribute

```
on_the_5_minute = 'on the 5 minute'
```

on_the_5_second class-attribute

```
on_the_5_second = 'on the 5 second'
```

schedule class-attribute

```
schedule = 'schedule'
```

MSATimers

Class to create dictionary of timers for use in MSAScheduler.

Attributes

timer_jobs instance-attribute

```
timer_jobs = {
    MSATimerEnum.every_poll: [],
    MSATimerEnum.every_second: [],
    MSATimerEnum.on_the_5_second: [],
    MSATimerEnum.on_the_15_second: [],
    MSATimerEnum.on_the_30_second: [],
```

```
MSATimerEnum.every_minute: [],
MSATimerEnum.on_the_5_minute: [],
MSATimerEnum.on_the_15_minute: [],
MSATimerEnum.on_the_30_minute: [],
MSATimerEnum.every_hour: [],
MSATimerEnum.schedule: [],
}
```

Functions

`__init__`

```
__init__()
```

`self.timer_jobs` is the primary resource in `MSATimers`. This is filled by `MSATimers`. It is then accessed by the source and served to `MSAScheduler`.

`create_timer`

```
create_timer(
    T_mode: MSATimerEnum,
    func: typing.Callable,
    mark_HH_MM: str = None,
)
```

Create a Timer

PARAMETER	DESCRIPTION
<code>T_mode</code>	MSATimerEnum TYPE: <code>MSATimerEnum</code>
<code>func</code>	the call handler for this timer TYPE: <code>typing.Callable</code>
<code>mark_HH_MM</code>	If scheduler type then this is the time for execution. TYPE: <code>str</code> DEFAULT: <code>None</code>

Functions

`get_time`

```
get_time(local_time_zone = 'UTC')
```

get_time_stamp

```
get_time_stamp(  
    local_time_zone="UTC", time_format="HMS"  
)
```

Last update: September 13, 2022

Created: September 13, 2022