

## u2d\_msa\_sdk Module

### **.service**

---

Main Service Module for MSAApp.

Initialize with a MSAServiceDefintion Instance to control the features and functions of the MSAApp.

## Attributes

**\_\_version\_\_** module-attribute

```
__version__ = '0.0.3'
```

str: Module Version

**password\_helper** module-attribute

```
password_helper = PasswordHelper(security_context)
```

Password Helper Instance

**security** module-attribute

```
security = getMSASecurity()
```

MSASecurity instance

**security\_context** module-attribute

```
security_context = CryptContext(  
    schemes=["bcrypt"], deprecated="auto"  
)
```

Security Context for Password Helper

## Classes

### MSAApp

Bases: `MSAFastAPI`

Creates an application MSA SDK instance.



#### Note



As with FastApi the MSAApp provides two events: `startup`: A list of callables to run on application startup. Startup handler callables do not take any arguments, and may be either standard functions, or async functions. `shutdown`: A list of callables to run on application shutdown. Shutdown handler callables do not take any arguments, and may be either standard functions, or async functions. Those are also used internally, which are triggered before the external events.

Do not include the `self` parameter in the `Args` section.

PARAMETER	DESCRIPTION
<code>settings</code>	MSAServiceDefinition (Must be provided), instance of a service definition with all settings <b>TYPE:</b> <code>MSAServiceDefinition</code>
<code>timers</code>	MSATimers instance Default None, provide a MSATimers instance and it will start the scheduler internally <b>TYPE:</b> <code>MSATimers</code> <b>DEFAULT:</b> <code>None</code>
<code>sql_models</code>	List of SQLAlchemy Model Default None, provide list of your SQLAlchemy Model Classes and the instance can create CRUD API and if site is enabled also UI for CRUD <b>TYPE:</b> <code>List[SQLModel]</code> <b>DEFAULT:</b> <code>None</code>
<code>auto_mount_site</code>	Default True, if site is enabled in settings and this is true, mounts the site in internal startup event. <b>TYPE:</b> <code>bool</code> <b>DEFAULT:</b> <code>True</code>

ATTRIBUTE	DESCRIPTION
<code>logger</code>	loguru logger instance

ATTRIBUTE	DESCRIPTION
<code>auto_mount_site</code>	bool auto_mount_site <b>TYPE:</b> <code>bool</code>
<code>settings</code>	MSAServiceDefinition settings instance.
<code>timers</code>	MSATimers = timers <b>TYPE:</b> <code>MSATimers</code>
<code>healthdefinition</code>	MSAHealthDefinition settings.healthdefinition <b>TYPE:</b> <code>MSAHealthDefinition</code>
<code>limiter</code>	Limiter = None <b>TYPE:</b> <code>Limiter</code>
<code>db_engine</code>	AsyncEngine = Db Engine instance <b>TYPE:</b> <code>AsyncEngine</code>
<code>sql_models</code>	List[SQLModel] = sql_models <b>TYPE:</b> <code>List[SQLModel]</code>
<code>sql_cruds</code>	List[MSASQLModelCrud] = [] <b>TYPE:</b> <code>List[MSASQLModelCrud]</code>
<code>scheduler</code>	MSAScheduler = None <b>TYPE:</b> <code>MSAScheduler</code>
<code>site</code>	AdminSite Admin/Auth Site instance.
<code>scheduler_task</code>	The Task instance that runs the Scheduler in the Background <b>TYPE:</b> <code>Task</code>
<code>ROOTPATH</code>	str os.path.join(os.path.dirname( <b>file</b> ))

## Attributes

Base `instance-attribute`

```
Base: DeclarativeMeta = declarative_base()
```

**ROOTPATH** instance-attribute

```
ROOTPATH = os.path.join(os.path.dirname(__file__))
```

**auto\_mount\_site** instance-attribute

```
auto_mount_site: bool = auto_mount_site
```

**db\_engine** instance-attribute

```
db_engine: AsyncEngine = None
```

**graphql\_app** instance-attribute

```
graphql_app: GraphQLRouter = None
```

**graphql\_schema** instance-attribute

```
graphql_schema: schema = None
```

**healthcheck** instance-attribute

```
healthcheck: health.MSAHealthCheck = None
```

**healthdefinition** instance-attribute

```
healthdefinition: MSAHealthDefinition = (  
    self.settings.healthdefinition  
)
```

**limiter** instance-attribute

```
limiter: Limiter = None
```

**logger** instance-attribute

```
logger = logger
```

**scheduler** instance-attribute

```
scheduler: MSAScheduler = None
```

scheduler\_task instance-attribute

```
scheduler_task: Task = None
```

settings instance-attribute

```
settings = settings
```

site instance-attribute

```
site = None
```

sql\_cruds instance-attribute

```
sql_cruds: List[MSASQLModelCrud] = []
```

sql\_models instance-attribute

```
sql_models: List[SQLModel] = sql_models
```

templates instance-attribute

```
templates = Jinja2Templates(
    directory=self.settings.templates_dir
)
```

timers instance-attribute

```
timers: MSATimers = timers
```

## Functions

`__init__`

```
__init__(
    settings: MSAServiceDefinition,
    timers: MSATimers = None,
    sql_models: List[SQLModel] = None,
    auto_mount_site: bool = True,
    *args,
```

```
    **kwargs
) -> None
```

### **get\_healthcheck** async

```
get_healthcheck(request: Request) -> ORJSONResponse
```

Get Healthcheck Status

### **get\_scheduler\_status** async

```
get_scheduler_status(
    request: Request,
) -> MSASchedulerStatus
```

Get Service Status Info

### **get\_services\_definition**

```
get_services_definition(
    request: Request,
) -> MSAServiceDefinition
```

Get Service Definition Info

### **get\_services\_openapi\_info**

```
get_services_openapi_info(
    request: Request,
) -> MSAOpenAPIInfo
```

Get Service OpenAPI Info

### **get\_services\_openapi\_schema**

```
get_services_openapi_schema(
    request: Request,
) -> ORJSONResponse
```

Get Service OpenAPI Schema

### **get\_services\_settings**

```
get_services_settings(request: Request) -> ORJSONResponse
```

## Get Service OpenAPI Schema

### **get\_services\_status** async

```
get_services_status(request: Request) -> MSAServiceStatus
```

## Get Service Status Info

### **index\_page**

```
index_page(request: Request) -> _TemplateResponse
```

## Get Service Index.html Page

### **monitor** async

```
monitor(request: Request) -> _TemplateResponse
```

Simple Service Monitor Page. Only works if pages is enabled in MSAServiceDefinition :param request: :return:

### **monitor\_inline** async

```
monitor_inline(request: Request) -> _TemplateResponse
```

Simple Monitor Page as Inline without head and body tags. Only works if pages is enabled in MSAServiceDefinition :param request: :return:

### **mount\_site**

```
mount_site() -> None
```

### **msa\_exception\_handler** async

```
msa_exception_handler(request: Request, exc: HTTPException)
```

Handles all HTTPExceptions if enabled with HTML Response or forward error if the code is in the exclude settings list. :param request: :type request: :param exc: :type exc: :return: :rtype:

### **msa\_exception\_handler\_disabled** async

```
msa_exception_handler_disabled(
    request: Request, exc: HTTPException
) -> JSONResponse
```

Handles all HTTPExceptions if Disabled with JSON Response. :param request: :type request:  
:param exc: :type exc: :return: :rtype:

## profiler

```
profiler(request: Request) -> _TemplateResponse
```

Simple Profiler Page. Only works if pages is enabled in MSAServiceDefinition :param request:  
:return:

## shutdown\_event async

```
shutdown_event() -> None
```

## startup\_event async

```
startup_event() -> None
```

:return: :rtype:

## testpage

```
testpage(request: Request) -> _TemplateResponse
```

Simple Testpage to see if the Micro Service is up and running. Only works if pages is enabled in MSAServiceDefinition :param request: :return:

## validation\_exception\_handler async

```
validation_exception_handler(
    request: Request, exc: RequestValidationError
) -> JSONResponse
```

# MSAOpenAPIInfo

Bases: SQLModel

**MSAOpenAPIInfo** Pydantic Response Class

## Attributes

name class-attribute



```
name: str = 'MSA SDK Service'
```

Service Name.

tags class-attribute

```
tags: Optional[List[str]] = None
```

OpenAPI Tags.

url class-attribute

```
url: str = '/openapi.json'
```

OpenAPI URL.

version class-attribute

```
version: str = '0.0.0'
```

API Version.

## MSASchedulerStatus

Bases: SQLModel

**MSASchedulerStatus** Pydantic Response Class

### Attributes

message class-attribute

```
message: Optional[str] = 'None'
```

Optional Message Text

name class-attribute

```
name: Optional[str] = 'MSA SDK Service'
```

Service Name.

timers class-attribute

```
timers: Optional[List[MSATimerStatus]] = []
```

Optional MSATimerStatus List

## MSAServiceStatus

Bases: `SQLModel`

**MSAServiceStatus** Pydantic Response Class

### Attributes

healthy `class-attribute`

```
healthy: Optional[str] = 'None'
```

Health status

message `class-attribute`

```
message: Optional[str] = 'None'
```

Optional Message Text

name `class-attribute`

```
name: Optional[str] = 'MSA SDK Service'
```

Service Name.

## MSATimerStatus

Bases: `SQLModel`

**MSATimerStatus** Pydantic Response Class

### Attributes

func `class-attribute`

```
func: Optional[str] = None
```

Timer Handler Function.

mark\_HH\_MM class-attribute

```
mark_HH_MM: Optional[str] = None
```

Mark for Schedule

mode class-attribute

```
mode: Optional[str] = None
```

Timer Mode.

## Functions

### getSecretKey

```
getSecretKey()
```

Get Secret Key for Token creation from OS Environment Variable **SECRET\_KEY\_TOKEN**

RETURNS	DESCRIPTION
key	The SECRET_KEY_TOKEN.

### getSecretKeyCSRF

```
getSecretKeyCSRF() -> str
```

Get Secret Key for CSRF Middleware from OS Environment Variable **SECRET\_KEY\_CSRF**

RETURNS	DESCRIPTION
key	The SECRET_KEY_CSRF. <b>TYPE:</b> str

### getSecretKeySessions

```
getSecretKeySessions()
```

## Get Secret Key for Session Middleware from OS Environment Variable **SECRET\_KEY\_SESSIONS**

RETURNS	DESCRIPTION
<code>key</code>	The SECRET_KEY_SESSIONS.

Last update: September 13, 2022

Created: September 13, 2022