TGF-SW-BASESTATION Documentation

None

MeshPower LTD

Table of contents

1. Introduction

2. API Reference

2.1 Devices

2.1.1 devices.api

Classes

DEVICEAPI

Bases: `Protocol`

Protocol defining the interface for device API implementations.

Classes implementing this protocol must provide methods to interact with device APIs, specifically for retrieving device variables.

Source code in `devices/api.py`

```
53  class DeviceApi(Protocol):
54      """Protocol defining the interface for device API implementations.
55
56      Classes implementing this protocol must provide methods to interact with
57      device APIs, specifically for retrieving device variables.
58      """
59
60      def get_vars(self, device_id: str) -> None:
61          """Retrieve variables for a specific device.
62
63          Args:
64              device_id: The unique identifier of the device.
65
66          Returns:
67              None
68          """
69          ...
```

`get_vars(device_id: str) -> None`

Retrieve variables for a specific device.

Parameters:

Name	Type	Description	Default
`device_id`	`str`	The unique identifier of the device.	*required*

Returns:

Type	Description
`None`	None

Source code in `devices/api.py`

```
60  def get_vars(self, device_id: str) -> None:
61      """Retrieve variables for a specific device.
62
63      Args:
64          device_id: The unique identifier of the device.
65
66      Returns:
67          None
68      """
69      ...
```

Functions

REGISTER_DEVICE_API

Register a DeviceApi class to the factory registry.

This decorator registers a class that implements the DeviceApi protocol to the DEVICE_API_FACTORY dictionary, making it available for retrieval via `get_device_api()`.

Parameters:

Name	Type	Description	Default
`cls`		The DeviceApi implementation class to register.	*required*

Returns:

Type	Description
	The class unchanged, allowing it to be used as a decorator.

Source code in `devices/api.py`

```
def register_device_api(cls):
    """Register a DeviceApi class to the factory registry.

    This decorator registers a class that implements the DeviceApi protocol
    to the DEVICE_API_FACTORY dictionary, making it available for retrieval
    via `get_device_api()`.

    Args:
        cls: The DeviceApi implementation class to register.

    Returns:
        The class unchanged, allowing it to be used as a decorator.
    """
    logger.warning(f"Registering {cls} to device api factory")
    DEVICE_API_FACTORY[cls.__name__] = cls
    return cls
```

GET_DEVICE_API

Retrieve a DeviceApi class from the factory registry.

Parameters:

Name	Type	Description	Default
`api_name`	`str`	The name of the DeviceApi class to retrieve.	*required*

Returns:

Type	Description
`DeviceApi | None`	The DeviceApi class if found, None otherwise. Logs an error
`DeviceApi | None`	if the API name is not found in the registry.

Source code in `devices/api.py`

```python
def get_device_api(api_name: str) -> "DeviceApi | None":
    """Retrieve a DeviceApi class from the factory registry.

    Args:
        api_name: The name of the DeviceApi class to retrieve.

    Returns:
        The DeviceApi class if found, None otherwise. Logs an error
        if the API name is not found in the registry.
    """
    api = DEVICE_API_FACTORY.get(api_name)
    if not api:
        logger.error(f"api {api_name} not found in {DEVICE_API_FACTORY=}")
    return api
```

Variables

`devices.api.DEVICE_API_FACTORY: dict[str, DeviceApi] = {}` `module-attribute`

Factory registry for DeviceApi implementations.

Maps API class names to their corresponding DeviceApi class implementations. This dictionary is populated by the `register_device_api` decorator.

2.1.2 devices.device

Classes

DEVICE

Base class for device implementations.

This class provides the core functionality for device communication, variable polling, state management, and data persistence. Subclasses should override `_get_vars()` to implement device-specific variable retrieval logic.

Attributes:

Name	Type	Description
`DEVICE_MATCH`	`list[str]`	List of device type identifiers used for factory registration.
`DEVICE_STATE_OBJECTS`		List of attribute names that should be included in device state serialization.

Source code in `devices/device.py`