WolkConnect-Python

Release 5.1.0

WolkAbout

Jun 05, 2023

TABLE OF CONTENT

Welcome to WolkConnect-Python's documentation!

CHAPTER

ONE

WOLKCONNECT LIBRARY

WolkConnect libraries are used to enable a device's communication with WolkAbout IoT platform instance. Using WolkConnect libraries in the software or firmware of a device will drastically decrease the time to market for developers or anyone wanting to integrate their own product with WolkAbout IoT Platform.

WolkConnect libraries are intended to be used on IP enabled devices. The available WolkConnect libraries (implemented in the following programming languages C, C++, Java, Python, Node-RED) are platform independent for OS based devices, with a special note that the WolkConnect-C library is suitable to be adapted for the use on non-OS devices as WolkConnect libraries have a small memory footprint. More hardware specific WolkConnect libraries are available for Arduino, MicroPython and Zerynth.

Features of WolkAbout IoT Platform that have been incorporated into WolkConnect libraries will be disambiguated with information on how to perform these features on devices by using WolkConnect's API.

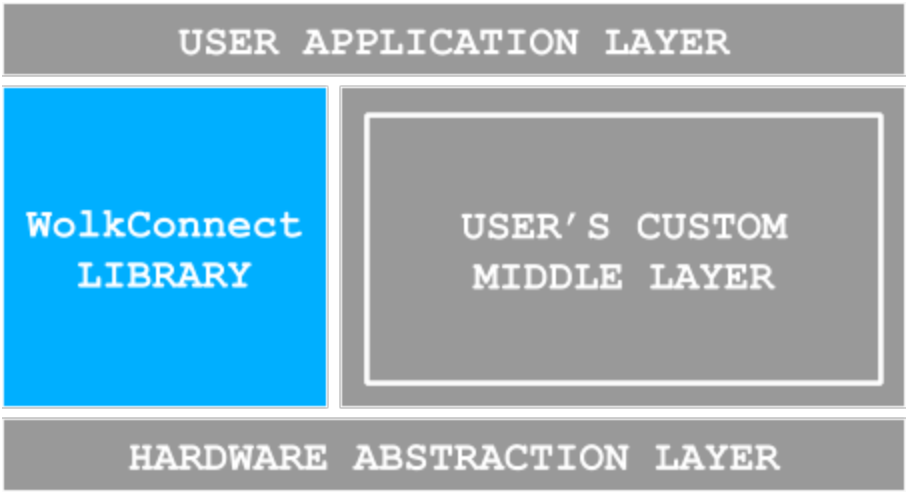
WolkConnect libraries are open-source and released under the Apache License 2.0.

1.1 Architecture

WolkConnect library is intended to be used as a dependency in other firmware or software that have their own existing business logic. WolkConnect library is not, by any means, a single service to control the device, it is a library intended to handle all the specific communication with WolkAbout IoT Platform.

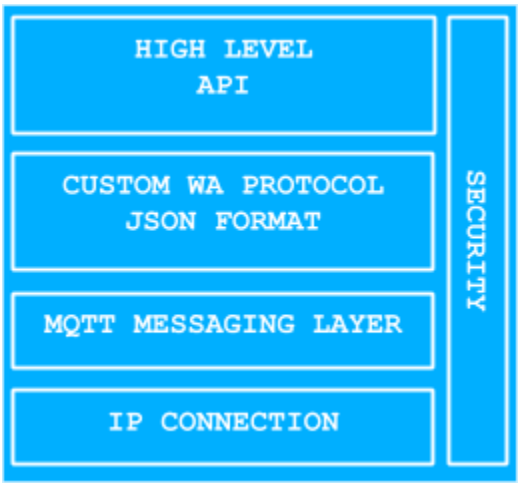
Using a WolkConnect library requires minimal knowledge of WolkAbout IoT Platform, no knowledge of the internal mechanisms and protocols of WolkAbout IoT Platform is necessary. The user only utilizes APIs provided by WolkConnect library in the User Application Layer, thereby reducing time-to-market required.

The architecture of software/firmware where WolkConnect library is meant to be used is presented in *Fig.1.1*. The gray section in *Fig.1.1* represents the developer's software/firmware.



The gray section between the User Application Layer and the Hardware Abstraction Layer represents the user's libraries and drivers that are required for his project. Providing WolkConnect library with IP connectivity from the Hardware Abstraction Layer is expected from the user.

WolkConnect library is separated into layers as shown in *Fig.1.2*



WolkConnect libraries use IP connectivity provided by the OS, but on devices where this is not available, it is the user's responsibility to provide implementations for opening a socket and send/receive methods to the socket.

Communication between WolkConnect library and WolkAbout IoT Platform is achieved through the use of the MQTT messaging protocol. WolkConnect libraries have a common dependency, an implementation of an MQTT client that will exchange data with an MQTT server that is part of WolkAbout IoT Platform. The communication between WolkConnect library and WolkAbout IoT Platform is made secure with the use of Secure Sockets Layer (SSL) if the device and MQTT client library support it.

Another common dependency for WolkConnect libraries is a JSON library that is used for parsing data that is exchanged with WolkAbout IoT Platform. This data is formatted using a custom JSON based protocol defined by WolkAbout IoT Platform.

The high-level API represents what is available to the developer that is using WolkConnect library. APIs follow the naming convention of the programming language they were written in. Consult a specific WolkConnect library's documentation for more information. The API is divided into three parts: connection management, data handling and device management. Data handling is independent of device management on WolkAbout IoT Platform and therefore has a separate API. Device management is responsible for device health and this, in turn, increases the device's lifespan.

1.2 API's functional description

WolkConnect libraries separate device's functionality through the API into three distinct parts:

- **Connection Management** - allows controlling the connected device in order to maintain data delivery integrity:
  - Connect
  - Disconnect
- **Data Handling** - valuable data to be exchanged with WolkAbout IoT Platform:
  - Feed values
  - Timestamp request
- **Device management** - dynamical modification of the device properties with the goal to change device behavior:
  - Feed registration
  - Feed removal
  - Attribute registration
  - File Management
  - Device Software/Firmware Update

1.2.1 Connection Management

Every connection from WolkConnect library to WolkAbout IoT Platform is authenticated with a device key and a device password. These credentials are created on WolkAbout IoT Platform when a device is created and are unique to that device. Only one active connection is allowed per device.

Attempting to create an additional connection with the same device credentials will terminate the previous connection. The connection is made secure, by default, in all WolkConnect libraries through the use of Secure Sockets Layer (SSL). Connecting without SSL is possible. For more information, refer to specific WolkConnect library documentation.

Connect

A device can be connected to WolkAbout IoT Platform in two ways:

- **Always connected devices** - connect once and publish data when necessary. This is a device that has a data delivery type of `PUSH`. Feed values or other commands for the device are issued instantly to the device when it is connected.
- **Periodically connected devices** - connect and publish data when needed. This dis a device that has a data delivery type of `PULL`. All pending commands from the Platform are polled by calling appropriate pull functions upon establishing connection.

Disconnect

Disconnecting will gracefully terminate the connection to WolkAbout IoT Platform.

1.2.2 Data Handling

Feed Values

Information needs to be distinguishable, so every piece of data sent from the device needs to have an identifier. This identifier is called a reference, and all the references of a device on WolkAbout IoT Platform must be unique.

Real world devices can perform a wide variety of operations that result in meaningful data. These operations could be to conduct a measurement, monitor certain conditions or execute some form of command. The data resulting from these operations have been modeled into two distinct types of device feeds:

- `In` feed - where data is only published from the device to the Platform.
- `In/Out` feed - where data is published in both directions, where the Platform can request that a feed be set to a specified value every time the ping keep-alive mechanism receives a response to its message.

Timestamp Request

Some devices might need a timestamp to perform some actions and they can issue a request from the Platform for the current Unix epoch time

1.2.3 Device Management

Feed Registration

The device is able to register new feeds for itself and immediately start publishing for that feed. The registration request consists of the specified feed name (displayed on the Platform), reference, feed type (In or In/Out) and the measurement unit. The new feed needs to have per-device unique reference The measurement unit can be one that is by default provided by the Platform or defined by a user.

Feed Removal

Issue a request that a specified feed, identified by reference, be removed from the device. Feed values for that reference will be discarded by the Platform after the feed removal request.

Attribute Registration

The device is able to register an attribute that better describes this specific device. The registration request contains a unique name (displayed on the Platform), the data type of the attribute (enumeration), as well as the value of the attribute, which is always sent as a string regardless of data type. If an attribute with the given name already exists, its value will be updated.

File Management

Devices that have the ability to store files in permanent memory can support the file management feature. This enables the Platform to transfer files to the device in pieces via MQTT or optionally, if the device supports it, tell the device to download a file from a URL. If the device supports this feature, it is expected that the device will publish a list of files present on it as soon as it establishes connection to the Platform. Apart from commands to transfer new files to the device, the Platform can also issue commands to delete one or all files present on the device.

Device Software/Firmware Update

WolkAbout IoT Platform gives the possibility of updating the device software/firmware. To enable this functionality, the device is required to have file management enabled. The process is separated into two autonomous stages:

- Start the process of installing a file on the device
- Verify installed software/firmware

The device needs to be connected to the Platform and deliver current software/firmware version to WolkAbout IoT Platform before starting to exploit utilize software/firmware update functionality.

WolkAbout IoT Platform actuates the device to start the process of installing. The responsibility to successfully install the file is on a device, not on WolkConnect library. In order to update the firmware, the user must create a firmware handler.

This firmware handler will specify the following parameters:

- Current firmware version,
- Implementation of a firmware installer that will be responsible for the installation process, as well as the possibility of a command to abort the installation process.

1.3 API Examples

To see how to utilize WolkConnect library APIs, explore some examples:

- Simple example - demonstrates the periodic sending of a temperature feed value
- Pull example - demonstrates the PULL data delivery type of device
- Register feed & attribute example - demonstrates the registration of a new device feed and device attribute
- Full feature set example - demonstrates all WolkConnect features.

CHAPTER

TWO

README

WolkAbout Python Connector library for connecting devices to WolkAbout IoT platform instance.

2.1 Prerequisite

- Python 3.7+

2.2 Installation

There are two ways to install this package

2.2.1 Installing with pip

```
python3 -m pip install wolk-connect
```

2.2.2 Installing from source

Clone this repository from the command line using:

```
git clone https://github.com/Wolkabout/WolkConnect-Python.git
```

Install dependencies by invoking `python3 -m pip install -r requirements.txt`

Install the package by running:

```
python3 setup.py install
```

2.3 Example Usage

2.3.1 Establishing connection with WolkAbout IoT platform

Create a device on WolkAbout IoT Platform by using the *Simple example* device type that is available on the platform. `Note that device type can be created by importing` `simple_example.json` `file as new Device Type.` This device type fits main.py and demonstrates the periodic sending of a temperature feed reading.

```
import wolk

# Setup the device credentials which you received
# when the device was created on the platform
device = wolk.Device(key="device_key", password="some_password")

# Pass your device and server information
# defaults to secure connection to Demo instance - comment out host, port and ca_cert
wolk_device = wolk.WolkConnect(
    device, host="insert_host", port=80, ca_cert="PATH/TO/YOUR/CA.CRT/FILE"
)

wolk_device.connect()
```

2.3.2 Adding feed values

```
wolk_device.add_feed_value(("T", 26.93))

# or multiple feed value readings
wolk_device.add_feed_value([("T", 27.11), ("H", 54.34), ("P", 1002.3)])
```

Optionally pass a `timestamp` as `round(time.time()) * 1000`. This is useful for maintaining data history when readings are not published immediately after adding them to storage. If `timestamp` is not provided, the library will assign a timestamp before placing the reading into storage.

Adding feed values with timestamp

```
# Add a signel feed reading to the message queue with the timestamp
wolk_device.add_feed_value(("T", 12.34), 1658315834000)

# Add a multi feed reading to the message queue with the timestamp
wolk_device.add_feed_value([("T", 12.34), ("H", 56.78), ("P", 1022.00)], 1658315834000)
```

2.3.3 Readings persistence and limit

Readings with method `add_feed_value` are added into local persistence. When adding messages be mindful of the message size that will be published. The default MQTT message size is 260MB, and since readings are of different sizes (based on the users use-case), check that the limit of readings in persistence will be under the MQTT limit for your broker. The default readings limit is set to 500000. You can change it with `set_custom_readings_persistence_limit`, if your readings are bigger, you can decrease the size, or if you have smaller readings, you can increase the size.

2.3.4 Data publish strategy

Stored feed values are pushed to WolkAbout IoT platform on demand by calling:

```
wolk_device.publish()
```

2.3.5 Adding feed values 'separated'

When adding feed values, the values themselves are persisted, which means when publishing all values will be placed in a single message and published as a single message.

If you would like to ensure different behavior, where you can add feed values that will be sent as a separate message from any other feed values, use the alternative method:

```
# Method arguments are exactly the same as for the `add_feed_value`
wolk_device.add_feed_value_separated([("T", 12.34), ("H", 56.78), ("P", 1022.00)],␣
↪1658315834000)
```

2.3.6 Disconnecting from the platform

```
wolk_device.disconnect()
```

2.4 Additional functionality

WolkConnect-Python library has integrated additional features which can perform full WolkAbout IoT platform potential. Explore the examples for more information.

CHAPTER

THREE

WOLK PACKAGE

Module that provides connection to WolkAbout IoT Platform.

To start publishing data to the platform create an instance of Device class with credentials obtained from the platform and pass it to an instance of WolkConnect class.

For more information about module features visit: https://github.com/Wolkabout/WolkConnect-Python/tree/master/examples/full_feature_set

3.1 WolkConnect

Core of this package. Wraps in all functionality.

class wolk.wolk_connect.**WolkConnect**(*device:* Device, *host: Optional[str] = None*, *port: Optional[int] = None*, *ca_cert: Optional[str] = None*)

Bases: object

Exchange data with WolkAbout IoT Platform.

Variables

- **connectivity_service** (ConnectivityService) – Means of sending/receiving data
- **device** (Device) – Contains device key and password
- **file_management** (FileManagement *or None*) – File management module
- **firmware_update** (FirmwareUpdate *or None*) – Firmware update handler
- **logger** (*logging.Logger*) – Logger instance issued by wolk.LoggerFactory
- **message_deserializer** (MessageDeserializer) – Deserializer of inbound messages
- **message_factory** (MessageFactory) – Create messages to send
- **message_queue** (MessageQueue) – Store data before sending
- **readings_persistence** (*ReadingsPersistence*) – Store readings before sending
- **readings_limit** (*int*) – Limit of readings stored in persistence

add_feed_value(*reading: Union[Tuple[str, Union[bool, int, float, str]], List[Tuple[str, Union[bool, int, float, str]]]]*, *timestamp: Optional[int] = None*) → None

Place a feed value reading into storage.

A reading is identified by a unique feed reference string and the current value of the feed.

This reading can either be passed as a tuple of (reference, value) for a single feed or as a list of previously mentioned tuples to pass multiple feed readings at once.

A Unix epoch timestamp in milliseconds as int can be provided to denote when the reading occurred. By default, the current system provided time will be assigned to a reading.

Parameters

- **reading** (*Union[Reading, List[Reading]]*) – Feed value reading
- **timestamp** (*Optional[int]*) – Unix timestamp. Defaults to system time.

add_feed_value_separated(*reading: Union[Tuple[str, Union[bool, int, float, str]], List[Tuple[str, Union[bool, int, float, str]]]]*, *timestamp: Optional[int] = None*) → None

Place a feed value reading into storage.

A reading is identified by a unique feed reference string and the current value of the feed.

This reading can either be passed as a tuple of (reference, value) for a single feed or as a list of previously mentioned tuples to pass multiple feed readings at once.

A Unix epoch timestamp in milliseconds as int can be provided to denote when the reading occurred. By default, the current system provided time will be assigned to a reading.

The separated variant will ensure that these reading values get sent as a separate message, independent of any other feed values that have been added to the object.

Parameters

- **reading** (*Union[Reading, List[Reading]]*) – Feed value reading
- **timestamp** (*Optional[int]*) – Unix timestamp. Defaults to system time.

connect() → None

Connect the device to the WolkAbout IoT Platform.

If the connection is made, then it also sends information about list of files present on device, current firmware version and the result of the firmware update process.

disconnect() → None

Disconnect the device from WolkAbout IoT Platform.

publish() → None

Publish all currently stored messages to WolkAbout IoT Platform.

pull_feed_values() → None

Issue a message to pull commanded feed values.

pull_parameters() → None

Issue a message to pull commanded feed values.

register_attribute(*name: str*, *data_type:* DataType, *value: str*) → None

Register an attribute for the device.

The attribute name must be unique per device. All attributes created by a device are always required and read-only. If an attribute with the given name already exists, the value will be updated.

Parameters

- **name** (*str*) – Unique attribute name
- **data_type** (DataType) – Data type this attribute will hold
- **value** (*str*) – Value of the attribute

register_feed(*name: str*, *reference: str*, *feed_type:* FeedType, *unit: Union[*Unit, *str]*) → None

Register a new feed for the device.

Feed is identified by name, unique reference, type (in or in/out) and unit; Where unit is either a default available unit listed in Unit enumeration, or a custom user defined unit that should be passed as a string value.

Parameters

- **name** (*str*) – Feed name
- **reference** (*str*) – Unique identifier
- **feed_type** (FeedType) – Is the feed one or two-way communication
- **unit** (*Union[Unit, str]*) – Unit used to measure this feed

remove_feed(*reference: str*) → None

Remove a feed from the device.

Parameters
reference (*str*) – Unique identifier

request_timestamp() → Optional[int]

Return last received timestamp from Platform.

If the device didn't connect at least once, then this will return None.

Returns
UTC timestamp in milliseconds

Return type
int or None

set_custom_readings_persistence_limit(*limit: int*)

Change the limit for readings persistence.

Parameters
limit (*int*) – New limit for readings persistence

with_custom_connectivity(*connectivity_service:* ConnectivityService)

Provide a custom way to communicate with the Platform.

Parameters
connectivity_service (ConnectivityService) – Custom connectivity service

with_custom_message_queue(*message_queue:* MessageQueue)

Use custom means of storing serialized messages.

Parameters
message_queue (MessageQueue) – Custom message queue

with_custom_protocol(*message_factory:* MessageFactory, *message_deserializer:* MessageDeserializer)

Provide a custom protocol to use for communication with the Platform.

Parameters

- **message_factory** (MessageFactory) – Creator of messages to be sent to the Platform
- **message_deserializer** (MessageDeserializer) – Deserializer of messages from the Platform

with_custom_readings_persistence(*readings_persistence: ReadingsPersistence*)

Use custom means of storing readings.

Parameters
readings_persistence (*ReadingsPersistence*) – Custom readings persistence

with_file_management(*file_directory: str*, *preferred_package_size: int = 0*, *url_downloader: Optional[Callable[[str, str], bool]] = None*)

Enable file management on the device.

Parameters

- **file_directory** (*str*) – Directory where files are stored
- **preferred_package_size** (*int*) – Size in kilobytes, 0 means no limit
- **url_downloader** (*Optional[Callable[[str, str], bool]]*) – Function for downloading file from URL

with_firmware_update(*firmware_handler:* FirmwareHandler)

Enable firmware update for device.

Requires that file management is previously enabled on device.

Parameters
firmware_handler (FirmwareHandler) – Provide firmware version & handle installation

with_incoming_feed_value_handler(*incoming_feed_value_handler: Callable[[List[Dict[str, Union[bool, int, float, str]]]], None]*)

Enable device to respond to incoming feed value change commands.

Commands will be delivered as a list of dictionaries that contain a feed_reference:value pair, and a "timestamp" field that specifies when this command was issued from the platform. The timestamp is an int representing Unix milliseconds.

Parameters
incoming_feed_value_handler (*Callable[[IncomingData], None]*) – Handler of feed value commands

3.2 Connection Management

3.2.1 MQTT connectivity service

Connectivity service based on MQTT protocol.

class wolk.mqtt_connectivity_service.**MQTTConnectivityService**(*device:* Device, *topics: List[str]*, *qos: int = 2*, *host: str = 'insert_host'*, *port: int = 80*, *max_retries: int = 3*, *ca_cert: Optional[str] = None*)

Bases: *ConnectivityService*

Handle sending and receiving MQTT messages.

connect() → bool

Establish the connection to the WolkAbout IoT platform.

Subscribes to all topics defined by device communication protocol. Starts a loop to handle inbound messages.

Returns
Connection state, True if connected, False otherwise

Return type
bool

disconnect() → None

Disconnects the device from the WolkAbout IoT Platform.

is_connected() → bool

Return current connection state.

Returns
connected

Return type
bool

publish(*message:* Message) → bool

Publish serialized data to WolkAbout IoT Platform.

Parameters
message (Message) – Message to be published

Returns
result

Return type
bool

set_inbound_message_listener(*listener: Callable[[*Message*], None]*) → None

Set the callback function to handle inbound messages.

Parameters
listener (*Callable[[*Message*], None]*) – Function that handles inbound messages

3.3 Data Handling

3.3.1 WolkAbout Protocol message deserializer

Deserialize messages received in WolkAbout Protocol format.

class wolk.wolkabout_protocol_message_deserializer.**WolkAboutProtocolMessageDeserializer**(*device:* Device)

Bases: *MessageDeserializer*

Deserialize messages received from the WolkAbout IoT Platform.

Variables
logger (*logging.Logger*) – Logger instance issued by wolk.LoggerFactory

CHANNEL_DELIMITER = '/'

FEED_VALUES = 'feed_values'

FILE_BINARY = 'file_binary_response'

FILE_DELETE = 'file_delete'

FILE_LIST = 'file_list'

FILE_PURGE = 'file_purge'

FILE_UPLOAD_ABORT = 'file_upload_abort'

FILE_UPLOAD_INITIATE = 'file_upload_initiate'

FILE_URL_ABORT = 'file_url_download_abort'

FILE_URL_INITIATE = 'file_url_download_initiate'

FIRMWARE_ABORT = 'firmware_update_abort'

FIRMWARE_INSTALL = 'firmware_update_install'

PARAMETERS = 'parameters'

PLATFORM_TO_DEVICE = 'p2d/'

TIME = 'time'

get_inbound_topics() → List[str]

Return list of inbound topics for device.

Returns
List of topics to subscribe to

Return type
List[str]

is_feed_values(*message:* Message) → bool

Check if message is for incoming feed values.

Parameters
message (Message) – The message received

Returns
is_feed_values

Return type
bool

is_file_binary_response(*message:* Message) → bool

Check if message is file binary message.

Parameters
message (Message) – The message received

Returns
file_binary

Return type
bool

is_file_delete_command(*message:* Message) → bool

Check if message if file delete command.

Parameters
message (Message) – The message received

Returns
file_delete_command

Return type
bool

`is_file_list`(*message:* Message) → bool

Check if message is file list request message.

Parameters
message (Message) – The message received

Returns
file_list

Return type
bool

`is_file_management_message`(*message:* Message) → bool

Check if message is any kind of file management related message.

Parameters
message (Message) – The message received

Returns
is_file_management_message

Return type
bool

`is_file_purge_command`(*message:* Message) → bool

Check if message if file purge command.

Parameters
message (Message) – The message received

Returns
file_purge_command

Return type
bool

`is_file_upload_abort`(*message:* Message) → bool

Check if message is file upload command.

Parameters
message (Message) – The message received

Returns
file_upload_abort_command

Return type
bool

`is_file_upload_initiate`(*message:* Message) → bool

Check if message is file upload command.

Parameters
message (Message) – The message received

Returns
file_upload_initiate

Return type
bool

`is_file_url_abort`(*message:* Message) → bool

Check if message is file URL download command.

Parameters
message (Message) – The message received

Returns
file_url_download_abort

Return type
bool

`is_file_url_initiate`(*message:* Message) → bool

Check if message is file URL download command.

Parameters
message (Message) – The message received

Returns
file_url_download_init

Return type
bool

`is_firmware_abort`(*message:* Message) → bool

Check if message is firmware update command.

Parameters
message (Message) – The message received

Returns
firmware_update_abort

Return type
bool

`is_firmware_install`(*message:* Message) → bool

Check if message is firmware update install command.

Parameters
message (Message) – The message received

Returns
firmware_update_install

Return type
bool

`is_firmware_message`(*message:* Message) → bool

Check if message is any kind of firmware related message.

Parameters
message (Message) – The message received

Returns
is_firmware_message

Return type
bool

is_parameters(*message:* Message) → bool

Check if message is for updating device parameters.

Parameters
message (Message) – The message received

Returns
is_parameters

Return type
bool

is_time_response(*message:* Message) → bool

Check if message is response to time request.

Parameters
message (Message) – The message received

Returns
is_time_response

Return type
bool

parse_feed_values(*message:* Message) → List[Dict[str, Union[bool, int, float, str]]]

Parse the incoming feed values message.

Parameters
message (Message) – The message received

Returns
feed_values

Return type
List[Dict[str, Union[bool, int, float, str]]]

parse_file_binary(*message:* Message) → *FileTransferPackage*

Parse the message into a file transfer package.

Parameters
message (Message) – The message received

Returns
file_transfer_package

Return type
FileTransferPackage

parse_file_delete_command(*message:* Message) → List[str]

Parse the message into a list of file names.

Parameters
message (Message) – The message received

Returns
file_name

Return type
List[str]

parse_file_initiate(*message:* Message) → Tuple[str, int, str]

Return file name, file size and file hash from message.

Parameters
message (Message) – The message received

Returns
(name, size, hash)

Return type
Tuple[str, int, str]

parse_file_url(*message:* Message) → str

Parse the message into a URL string.

Parameters
message (Message) – The message received

Returns
file_url

Return type
str

parse_firmware_install(*message:* Message) → str

Return file name from message.

Parameters
message (Message) – The message received

Returns
file_name

Return type
str

parse_parameters(*message:* Message) → Dict[str, Union[bool, int, float, str]]

Parse the incoming parameters message.

Parameters
message (Message) – The message received

Returns
parameters

Return type
Dict[str, Union[bool, int, float, str]]

parse_time_response(*message:* Message) → int

Parse the message into an UTC timestamp.

Parameters
message (Message) – The message received

Returns
timestamp

Return type
int

3.3.2 WolkAbout protocol message factory

Factory for serializing messages according to WolkAbout Protocol.

class wolk.wolkabout_protocol_message_factory.**WolkAboutProtocolMessageFactory**(*device_key: str*)

Bases: *MessageFactory*

Serialize messages to be sent to WolkAbout IoT Platform.

ATTRIBUTE_REGISTRATION = 'attribute_registration'

CHANNEL_DELIMITER = '/'

DEVICE_TO_PLATFORM = 'd2p/'

FEED_REGISTRATION = 'feed_registration'

FEED_REMOVAL = 'feed_removal'

FEED_VALUES = 'feed_values'

FILE_BINARY_REQUEST = 'file_binary_request'

FILE_LIST = 'file_list'

FILE_UPLOAD_STATUS = 'file_upload_status'

FILE_URL_DOWNLOAD_STATUS = 'file_url_download_status'

FIRMWARE_UPDATE_STATUS = 'firmware_update_status'

FIRMWARE_VERSION_UPDATE = 'firmware_version_update'

PARAMETERS = 'parameters'

PULL_FEED_VALUES = 'pull_feed_values'

PULL_PARAMETERS = 'pull_parameters'

TIME = 'time'

make_attribute_registration(*name: str*, *data_type:* DataType, *value: str*) → *Message*

Serialize request to register an attribute for the device.

Parameters

- **name** (*str*) – Unique identifier
- **data_type** (DataType) – Type of data this attribute holds
- **value** (*str*) – Value of the attribute

Returns
message

Return type
Message

make_feed_registration(*name: str*, *reference: str*, *feed_type:* FeedType, *unit: Union[*Unit, *str]*) → *Message*

Serialize request to register a feed on the Platform.

Parameters

- **name** (*str*) – Feed name
- **reference** (*str*) – Unique identifier
- **feed_type** (FeedType) – Is the feed one or two-way communication
- **unit** (*Union[Unit, str]*) – Unit used to measure this feed

Returns
message

Return type
Message

make_feed_removal(*reference: str*) → *Message*

Serialize request to remove a feed from the device on the Platform.

Parameters
reference (*str*) – Unique identifier

Returns
message

Return type
Message

make_from_feed_value(*reading: Union[Tuple[str, Union[bool, int, float, str]], List[Tuple[str, Union[bool, int, float, str]]]]*, *timestamp: Optional[int]*) → *Message*

Serialize feed value data.

Parameters

- **reading** (*Union[Reading, List[Reading]]*) – Feed value data as (reference, value) or list of tuple
- **timestamp** – Unix timestamp in ms. Default to current time if None

Raises
ValueError – Reading is invalid data type

Returns
message

Return type
Message

make_from_feed_values_collected(*collected_readings: Dict[int, Dict[str, Union[bool, int, float, str]]]*) → *Message*

Serialize feed values collected over time.

Parameters
collected_readings (*Dict[int, Dict[str, OutgoingDataTypes]]*) – The map of collected readings.

Returns
The message containing all data.

Return type
Message

make_from_file_list(*file_list: List[Dict[str, Union[str, int]]]*) → *Message*

Serialize list of files present on device.

Parameters
file_list (*List[Dict[str, Union[str, int]]]*) – Files present on device

Returns
message

Return type
Message

make_from_file_management_status(*status:* FileManagementStatus, *file_name: str*) → *Message*

Serialize device's current file management status.

Parameters

- **status** (FileManagementStatus) – Current file management status
- **file_name** (*str*) – Name of file being transferred

Returns
message

Return type
Message

make_from_file_url_status(*file_url: str*, *status:* FileManagementStatus, *file_name: Optional[str] = None*) → *Message*

Serialize device's current file URL download status.

Parameters

- **file_url** (*str*) – URL from where the file is to be downloaded
- **status** (FileManagementStatus) – Current file management status
- **file_name** (*Optional[str]*) – Only present when download of file is completed

make_from_firmware_update_status(*firmware_update_status:* FirmwareUpdateStatus) → *Message*

Serialize firmware update status to be sent to WolkAbout IoT Platform.

Parameters
firmware_update_status – Firmware update status to be serialized

Returns
message

Return type
Message

make_from_package_request(*file_name: str*, *chunk_index: int*) → *Message*

Request a package of the file from WolkAbout IoT Platform.

Parameters

- **file_name** (*str*) – Name of the file that contains the requested package
- **chunk_index** (*int*) – Index of the requested package

Returns
message

Return type
Message

make_from_parameters(*parameters: Dict[str, Union[bool, int, float, str]]*) → *Message*

Serialize device parameters to be sent to the Platform.

Parameters
parameters (*Dict[str, Union[bool, int, float, str]]*) – Device parameters

Returns
message

Return type
Message

make_pull_feed_values() → *Message*

Serialize message requesting any pending inbound feed values.

Returns
message

Return type
Message

make_pull_parameters() → *Message*

Serialize request to pull device parameters from the Platform.

Returns
message

Return type
Message

make_time_request() → *Message*

Serialize message requesting platform timestamp.

Returns
message

Return type
Message

3.3.3 Message deque

Message storage implemented via double ended queue.

class wolk.message_deque.**MessageDeque**

Bases: *MessageQueue*

Store messages before they are sent to the WolkAbout IoT Platform.

Variables

- **logger** (*logging.Logger*) – Logger instance issued by wolk.LoggerFactory
- **queue** (*collections.deque*) – Double ended queue used to store messages

get() → Optional[*Message*]

Take the first message from the queue.

Returns
message

Return type
Optional[*Message*]

peek() → Optional[*Message*]

Return the first message from the queue without removing it.

Returns
message

Return type
Optional[*Message*]

put(*message:* Message) → bool

Add the message to the queue.

Parameters
message (Message) – Message to place in the queue

Returns
success

Return type
bool

3.3.4 File management

OS File Management module.

class wolk.os_file_management.**OSFileManagement**(*status_callback: Callable[[str, FileManagementStatus], None], packet_request_callback: Callable[[str, int], None], url_status_callback: Callable[[str, FileManagementStatus, Optional[str]], None]*)

Bases: *FileManagement*

File transfer manager.

Enables device to transfer files from WolkAbout IoT Platform package by package or/and URL download as well as report list of files currently on device and delete them on request.

configure(*file_directory: str, preferred_package_size: int = 0*) → None

Configure options for file management module.

Parameters

- **file_directory** (*str*) – Path to where files are stored
- **preferred_package_size** (*int*) – Size in kilobytes, 0 means no limit

get_file_list() → List[Dict[str, Union[str, int]]]

Return list of files present on device.

Each list item is a dictionary that contains the name of the file, its size in bytes, and a MD5 checksum of the file.

Returns
file_list

Return type
List[Dict[str, Union[str, int]]]

get_file_path(*file_name: str*) → Optional[str]

Return path to file if it exists.

Parameters
file_name (*str*) – File for which to get path

Returns
file_path

Return type
Optional[str]

get_preffered_package_size() → int

Return preffered package size for file transfer.

Returns
preferred_package_size

Return type
int

handle_file_binary_response(*package:* FileTransferPackage) → None

Validate received package and store or use callback to request again.

Parameters
package (FileTransferPackage) – Package of file being transfered.

handle_file_delete(*file_names: List[str]*) → None

Delete files from device.

Parameters
file_names (*List[str]*) – Files to be deleted

handle_file_purge() → None

Delete all files from device.

handle_file_upload_abort() → None

Abort file upload and revert to idle status.

handle_file_url_download_abort() → None

Abort file URL download.

handle_file_url_download_initiation(*file_url: str*) → None

Start file transfer from specified URL.

Parameters
file_url (*str*) – URL from where to download file

handle_upload_initiation(*file_name: str*, *file_size: int*, *file_hash: str*) → None

Start making package requests and set status to file transfer.

Parameters

- **file_name** (*str*) – File name
- **file_size** (*int*) – Size in bytes
- **file_hash** (*str*) – MD5 hash of file

set_custom_url_downloader(*downloader: Callable[[str, str], bool]*) → None

Set the URL file downloader to a custom implementation.

Default implementation uses *requests* and is available as a static method within this class.

Parameters
downloader (*Callable[[Arg(str, 'file_url'), Arg(str, 'file_path')], bool]*) – Function that will download the file from the URL

supports_url_download() → bool

Return if the file management module supports URL download.

Returns
supports_url_download

Return type
bool

static url_download(*file_url: str*, *file_path: str*) → bool

Attempt to download file from specified URL.

Parameters

- **file_url** (*str*) – URL from which to download file
- **file_path** – Path where to store file

Type
file_path: str

Returns
Successful download

Return type
bool

3.3.5 Firmware update

Enables firmware update for device.

class wolk.os_firmware_update.**OSFirmwareUpdate**(*firmware_handler:* FirmwareHandler, *status_callback:* *Callable[[*FirmwareUpdateStatus*], None]*)

Bases: *FirmwareUpdate*

Responsible for everything related to the firmware update process.

get_current_version() → str

Return device's current firmware version.

Returns
Firmware version

Return type
str

handle_abort() → None

Handle the abort command received from the platform.

handle_install(*file_path: str*) → None

Handle received firmware installation command.

Parameters
file_path (*str*) – Firmware file to install

report_result() → None

Report the result of the firmware installation process.

3.4 Utility

3.4.1 Logger factory

LoggerFactory Module.

class wolk.logger_factory.**LoggerFactory**(*level=20*, *console=True*, *log_file=None*)

Bases: object

Factory for issuing ready to use loggers in other modules.

get_logger(*name: str*, *level: Optional[int] = None*) → Logger

Return a ready to use logger instance.

Parameters

- **name** (*str*) – Name of the logger
- **level** (*int or None*) – Override the log level

Returns
Logger instance

Return type
logger

set_device_key(*device_key: str*) → None

Set device key.

Parameters
device_key (*str*) – Device key

wolk.logger_factory.**logging_config**(*level: str*, *log_file: Optional[str] = None*) → None

Set desired log level and designate a log file.

Parameters

- **level** (*str*) – Available levels : debug, info, notset
- **log_file** (*str or None*) – path to log file

CHAPTER

FOUR

ABSTRACT BASE CLASSES AND FUNCTION STUBS

4.1 Connectivity service

Service for exchanging data with WolkAbout IoT Platform.

class wolk.interface.connectivity_service.**ConnectivityService**

Bases: ABC

Responsible for exchanging data with WolkAbout IoT Platform.

abstract connect() → bool

Connect to WolkAbout IoT Platform.

abstract disconnect() → None

Disconnect from WolkAbout IoT Platform.

abstract is_connected() → bool

Return current connection state.

Returns
connected

Return type
bool

abstract publish(*message:* Message) → bool

Publish a message to WolkAbout IoT Platform.

Parameters
outbound_message (Message) – Message to send

Returns
success

Return type
bool

abstract set_inbound_message_listener(*listener: Callable[[*Message*], None]*) → None

Set a callback method to handle inbound messages.

Parameters
listener (*Callable[[*Message*], None]*) – Method hat handles inbound messages

4.2 File management

Module responsible for handling files and file transfer.

class wolk.interface.file_management.**FileManagement**(*status_callback: Callable[[str, FileManagementStatus], None]*, *packet_request_callback: Callable[[str, int, int], None]*, *url_status_callback: Callable[[str, FileManagementStatus, Optional[str]], None]*)

Bases: ABC

File transfer manager.

Enables device to transfer files from WolkAbout IoT Platform
package by package or/and URL download.

abstract configure(*file_directory: str*, *preferred_package_size: int = 0*) → None

Configure options for file management module.

Parameters

- **file_directory** (*str*) – Path to where files are stored
- **preferred_package_size** (*int*) – Size in kilobytes, 0 means no limit

abstract get_file_list() → List[Dict[str, Union[str, int]]]

Return list of files present on device.

Each list item is a dictionary that contains the name of the file, its size in bytes, and a MD5 checksum of the file.

Returns
file_list

Return type
List[Dict[str, Union[str, int]]]

abstract get_file_path(*file_name: str*) → Optional[str]

Return path to file if it exists.

Parameters
file_name (*str*) – File for which to get path

Returns
file_path

Return type
Optional[str]

abstract get_preffered_package_size() → int

Return preffered package size for file transfer.

Returns
preferred_package_size

Return type
int

abstract handle_file_binary_response(*package: FileTransferPackage*) → None

Validate received package and store or use callback to request again.

Parameters
package (FileTransferPackage) – Package of file being transfered.

abstract handle_file_delete(*file_names: List[str]*) → None

Delete files from device.

Parameters
file_names (*List[str]*) – Files to be deleted

abstract handle_file_purge() → None

Delete all files from device.

abstract handle_file_upload_abort() → None

Abort file upload and revert to idle status.

abstract handle_file_url_download_abort() → None

Abort file URL download.

abstract handle_file_url_download_initiation(*file_url: str*) → None

Start file transfer from specified URL.

Parameters
file_url (*str*) – URL from where to download file

abstract handle_upload_initiation(*file_name: str*, *file_size: int*, *file_hash: str*) → None

Start making package requests and set status to file transfer.

Parameters

- **file_name** (*str*) – File name
- **file_size** (*int*) – Size in bytes
- **file_hash** (*str*) – base64 encoded sha256 hash of file

abstract set_custom_url_downloader(*downloader: Callable[[str, str], bool]*) → None

Set the URL file downloader to a custom implementation.

Parameters
downloader (*Callable[[str, str], bool]*) – Function that will download the file from the URL

abstract supports_url_download() → bool

Return if the file management module supports URL download.

Returns
supports_url_download

Return type
bool

4.3 Firmware handler

Firmware handler for file installation and version reporting.

class wolk.interface.firmware_handler.**FirmwareHandler**

Bases: ABC

Handle firmware installation and get current firmware version.

abstract get_current_version() → str

Obtain device's current firmware version.

Returns
version

Rtpe
str

abstract install_firmware(*firmware_file_path: str*) → None

Handle the installation of the firmware file.

Parameters
firmware_file_path (*str*) – Path where the firmware file is located

4.4 Firmware update

Enables firmware update for device.

class wolk.interface.firmware_update.**FirmwareUpdate**(*firmware_handler:* FirmwareHandler, *status_callback:* *Callable[[*FirmwareUpdateStatus*], None]*)

Bases: ABC

Firmware Update enabler.

Responsible for supervising firmware installation and reporting current firmware installation status and version.

abstract get_current_version() → str

Return device's current firmware version.

Returns
Firmware version

Return type
str

abstract handle_abort() → None

Handle received firmware installation abort command.

abstract handle_install(*file_path: str*) → None

Handle received firmware installation command.

Parameters
file_path (*str*) – Firmware file to install

abstract report_result() → None

Report the results of the firmware update process.

4.5 Message deserializer

Process messages received from WolkAbout IoT Platform.

class wolk.interface.message_deserializer.**MessageDeserializer**

Bases: ABC

Deserialize messages received from the platform.

abstract get_inbound_topics() → List[str]

Return list of inbound topics for device.

Returns
List of topics to subscribe to

Return type
List[str]

abstract is_feed_values(*message:* Message) → bool

Check if message is for incoming feed values.

Parameters
message (Message) – The message received

Returns
is_feed_values

Return type
bool

abstract is_file_binary_response(*message:* Message) → bool

Check if message is file binary message.

Parameters
message (Message) – The message received

Returns
file_binary_response

Return type
bool

abstract is_file_delete_command(*message:* Message) → bool

Check if message if file delete command.

Parameters
message (Message) – The message received

Returns
file_delete_command

Return type
bool

abstract is_file_list(*message:* Message) → bool

Check if message is file list request message.

Parameters
message (Message) – The message received

Returns
file_list

Return type
bool

abstract is_file_management_message(*message:* Message) → bool

Check if message is any kind of file management related message.

Parameters
message (Message) – The message received

Returns
is_file_management_message

Return type
bool

abstract is_file_purge_command(*message:* Message) → bool

Check if message if file purge command.

Parameters
message (Message) – The message received

Returns
file_purge_command

Return type
bool

abstract is_file_upload_abort(*message:* Message) → bool

Check if message is file upload command.

Parameters
message (Message) – The message received

Returns
file_upload_abort_command

Return type
bool

abstract is_file_upload_initiate(*message:* Message) → bool

Check if message is file upload command.

Parameters
message (Message) – The message received

Returns
file_upload_initiate_command

Return type
bool

abstract is_file_url_abort(*message:* Message) → bool

Check if message is file URL download command.

Parameters
message (Message) – The message received

Returns
file_url_download_abort

Return type
bool

abstract is_file_url_initiate(*message:* Message) → bool

Check if message is file URL download command.

Parameters
message (Message) – The message received

Returns
file_url_download_initiate

Return type
bool

abstract is_firmware_abort(*message:* Message) → bool

Check if message is firmware update command.

Parameters
message (Message) – The message received

Returns
firmware_update_abort_command

Return type
bool

abstract is_firmware_install(*message:* Message) → bool

Check if message is firmware update install command.

Parameters
message (Message) – The message received

Returns
firmware_update_install_command

Return type
bool

abstract is_firmware_message(*message:* Message) → bool

Check if message is any kind of firmware related message.

Parameters
message (Message) – The message received

Returns
is_firmware_message

Return type
bool

abstract is_parameters(*message:* Message) → bool

Check if message is for updating device parameters.

Parameters
message (Message) – The message received

Returns
is_parameters

Return type
bool

abstract is_time_response(*message:* Message) → bool

Check if message is response to time request.

Parameters
message (Message) – The message received

Returns
is_time_response

Return type
bool

abstract parse_feed_values(*message:* Message) → List[Dict[str, Union[bool, int, float, str]]]

Parse the incoming feed values message.

Parameters
message (Message) – The message received

Returns
feed_values

Return type
List[Dict[str, Union[bool, int, float, str]]]

abstract parse_file_binary(*message:* Message) → *FileTransferPackage*

Parse the message into a file transfer package.

Parameters
message (Message) – The message received

Returns
file_transfer_package

Return type
FileTransferPackage

abstract parse_file_delete_command(*message:* Message) → List[str]

Parse the message into a list of file names.

Parameters
message (Message) – The message received

Returns
file_name

Return type
List[str]

abstract parse_file_initiate(*message:* Message) → Tuple[str, int, str]

Return file name, file size and file hash from message.

Parameters
message (Message) – The message received

Returns
(file_name, file_size, file_hash)

Return type
Tuple[str, int, str]

abstract parse_file_url(*message:* Message) → str

Parse the message into a URL string.

Parameters
message (Message) – The message received

Returns file_url
Return type
str

abstract parse_firmware_install(*message:* Message) → str

Return file name from message.

Parameters
message (Message) – The message received

Returns
file_name

Return type
str

abstract parse_parameters(*message:* Message) → Dict[str, Union[bool, int, float, str]]

Parse the incoming parameters message.

Parameters
message (Message) – The message received

Returns
parameters

Return type
Dict[str, Union[bool, int, float, str]]

abstract parse_time_response(*message:* Message) → int

Parse the message into an UTC timestamp.

Parameters
message (Message) – The message received

Returns
timestamp

Return type
int

4.6 Message factory

Create messages from data that conform to device's specified protocol.

class wolk.interface.message_factory.**MessageFactory**

Bases: ABC

Serialize messages to be sent to WolkAbout IoT Platform.

abstract make_attribute_registration(*name: str*, *data_type:* DataType, *value: str*) → *Message*

Serialize request to register an attribute for the device.

Parameters

- **name** (*str*) – Unique identifier
- **data_type** (DataType) – Type of data this attribute holds
- **value** (*str*) – Value of the attribute

Returns
message

Return type
Message

abstract make_feed_registration(*name: str*, *reference: str*, *feed_type:* FeedType, *unit: Union[*Unit, *str]*) → *Message*

Serialize request to register a feed for the device on the Platform.

Parameters

- **name** (*str*) – Feed name
- **reference** (*str*) – Unique identifier
- **feed_type** (FeedType) – Is the feed one or two-way communication
- **unit** (*Union[Unit, str]*) – Unit used to measure this feed

Returns
message

Return type
Message

abstract make_feed_removal(*reference: str*) → *Message*

Serialize request to remove a feed from the device on the Platform.

Parameters
reference (*str*) – Unique identifier

Returns
message

Return type
Message

abstract make_from_feed_value(*reading: Union[Tuple[str, Union[bool, int, float, str]], List[Tuple[str, Union[bool, int, float, str]]]]*, *timestamp: Optional[int]*) → *Message*

Serialize feed value data.

Parameters

- **reading** (*Union[Reading, List[Reading]]*) – Feed value data as (reference, value) or list of tuple
- **timestamp** – Unix timestamp in ms. Default to current time if None

Returns
message

Return type
Message

abstract make_from_feed_values_collected(*collected_readings: Dict[int, Dict[str, Union[bool, int, float, str]]]*) → *Message*

Serialize feed values collected and organized by timestamp.

Parameters
collected_readings (*Dict[int, Dict[str, OutgoingDataTypes]]*) – Feed values, organized by timestamp, and then by reference.

Returns
The message containing all the data

Return type
Message

abstract make_from_file_list(*file_list: List[Dict[str, Union[str, int]]]*) → *Message*

Serialize list of files present on device.

Parameters
file_list (*List[Dict[str, Union[str, int]]]*) – Files present on device

Returns
message

Return type
Message

abstract make_from_file_management_status(*status:* FileManagementStatus, *file_name: str*) → *Message*

Serialize device's current file management status.

Parameters

- **status** (FileManagementStatus) – Current file management status
- **file_name** (*str*) – Name of file being transferred

Returns
message

Return type
Message

abstract make_from_file_url_status(*file_url: str*, *status:* FileManagementStatus, *file_name: Optional[str] = None*) → *Message*

Serialize device's current file URL download status.

Parameters

- **file_url** (*str*) – URL from where the file is to be downloaded
- **status** (FileManagementStatus) – Current file management status
- **file_name** (*Optional[str]*) – Only present when download of file is completed

abstract make_from_firmware_update_status(*firmware_update_status:* FirmwareUpdateStatus) → *Message*

Report the current status of the firmware update process.

Parameters
firmware_update_status (FirmwareUpdateStatus) – Status of the firmware update process

Returns
message

Return type
Message

abstract make_from_package_request(*file_name: str*, *chunk_index: int*) → *Message*

Request a package of the file from WolkAbout IoT Platform.

Parameters

- **file_name** (*str*) – Name of the file that contains the requested package
- **chunk_index** (*int*) – Index of the requested package

Returns
message

Return type
Message

abstract make_from_parameters(*parameters: Dict[str, Union[bool, int, float, str]]*) → *Message*

Serialize device parameters to be sent to the Platform.

Parameters
parameters (*Dict[str, Union[bool, int, float, str]]*) – Device parameters

Returns
message

Return type
Message

abstract make_pull_feed_values() → *Message*

Serialize message requesting any pending inbound feed values.

Returns
message

Return type
Message

abstract make_pull_parameters() → *Message*

Serialize request to pull device parameters from the Platform.

Returns
message

Return type
Message

abstract make_time_request() → *Message*

Serialize message requesting platform timestamp.

Returns
message

Return type
Message

4.7 Message queue

Store messages before sending them to WolkAbout IoT Platform.

class wolk.interface.message_queue.**MessageQueue**

Bases: ABC

Store messages on device before publishing to WolkAbout IoT Platform.

abstract get() → Optional[*Message*]

Get a message from storage.

Returns
message

Return type
Optional[*Message*]

abstract peek() → Optional[*Message*]

Get a message without removing from storage.

Returns
message

Return type
Optional[*Message*]

abstract put(*message:* Message) → bool

Place a message in storage.

Parameters
message (Message) – Message to be stored

Returns
result

Return type
bool

CHAPTER

FIVE

MODELS

5.1 Data Delivery

Enumeration of data delivery types.

class wolk.model.data_delivery.**DataDelivery**(*value*)

Bases: Enum

Enumeration of available data delivery types.

A device's data delivery mode is either an always connected device (PUSH), or a device that only periodically establishes connection and then subsequently checks if there are any pending messages that are intended for it (PULL).

PULL = 'PULL'

PUSH = 'PUSH'

5.2 Data Type

Enumeration of available data types on the Platform.

class wolk.model.data_type.**DataType**(*value*)

Bases: Enum

Enumeration of data types on the Platform.

BOOLEAN = 'BOOLEAN'

ENUM = 'ENUM'

HEXADECIMAL = 'HEXADECIMAL'

LOCATION = 'LOCATION'

NUMERIC = 'NUMERIC'

STRING = 'STRING'

5.3 Feed Type

Enumeration of feed types.

class wolk.model.feed_type.**FeedType**(*value*)

Bases: Enum

Enumeration of available feed types.

IN = 'IN'

IN_OUT = 'IN_OUT'

5.4 Device

Everything needed for authenticating a device on WolkAbout IoT Platform.

class wolk.model.device.**Device**(*key: str*, *password: str*, *data_delivery: Optional[DataDelivery] = DataDelivery.PUSH*)

Bases: object

Device identified by key and password, and its outbound data mode.

A device's data delivery mode is either an always connected device (PUSH), or a device that only periodically establishes connection and then subsequently checks if there are any pending messages that are intended for it (PULL).

Variables

- **key** (*str*) – Device's key
- **password** (*str*) – Device's unique password
- **data_delivery** (DataDelivery) – Is the device always connected or only periodically

data_delivery: Optional[*DataDelivery*] = 'PUSH'

key: str

password: str

5.5 File management error type

Enumeration of defined file management errors.

class wolk.model.file_management_error_type.**FileManagementErrorType**(*value*)

Bases: Enum

Enumeration of available file management errors.

FILE_HASH_MISMATCH = 'FILE_HASH_MISMATCH'

FILE_SYSTEM_ERROR = 'FILE_SYSTEM_ERROR'

MALFORMED_URL = 'MALFORMED_URL'

RETRY_COUNT_EXCEEDED = 'RETRY_COUNT_EXCEEDED'

TRANSFER_PROTOCOL_DISABLED = 'TRANSFER_PROTOCOL_DISABLED'

UNKNOWN = 'UNKNOWN'

UNSUPPORTED_FILE_SIZE = 'UNSUPPORTED_FILE_SIZE'

5.6 File management status

Information about the current status of the firmware update process.

class wolk.model.file_management_status.**FileManagementStatus**(*status:* FileManagementStatusType, *error: Optional[*FileManagementErrorType*] = None*)

Bases: object

Contains the status of the file management process.

Variables

- **status** – The status of the file management process
- **error** – The type of error that occurred

Ivartype status
FileManagementStatusType

Ivartype error
FileManagementErrorType or None

error: Optional[*FileManagementErrorType*] = None

status: *FileManagementStatusType*

5.7 File management status type

Statuses defined for file management.

class wolk.model.file_management_status_type.**FileManagementStatusType**(*value*)

Bases: Enum

Enumeration of available file management statuses.

ABORTED = 'ABORTED'

ERROR = 'ERROR'

FILE_READY = 'FILE_READY'

FILE_TRANSFER = 'FILE_TRANSFER'

5.8 File transfer package

File transfer package model.

class wolk.model.file_transfer_package.**FileTransferPackage**(*previous_hash: bytes*, *data: bytes*, *current_hash: bytes*)

Bases: object

Data from a file binary request response.

Variables

- **previous_hash** (*bytes*) – Hash of the previous chunk
- **data** (*bytes*) – Requested chunk
- **current_hash** (*bytes*) – Hash of the current chunk

current_hash: bytes

data: bytes

previous_hash: bytes

5.9 Firmware update error type

Firmware update error types.

class wolk.model.firmware_update_error_type.**FirmwareUpdateErrorType**(*value*)

Bases: Enum

Enumeration of possible firmware update errors.

INSTALLATION_FAILED = 'INSTALLATION_FAILED'

UNKNOWN = 'UNKNOWN'

UNKNOWN_FILE = 'UNKNOWN_FILE'

5.10 Firmware update status

Information about the current status of the firmware update process.

class wolk.model.firmware_update_status.**FirmwareUpdateStatus**(*status: FirmwareUpdateStatusType*, *error: Optional[FirmwareUpdateErrorType] = None*)

Bases: object

Contains the status of the firmware update process.

Variables

- **status** – The status of the firmware update process
- **error** – The type of error that occurred

Ivartype status
FirmwareUpdateStatusType

Ivartype error
FirmwareUpdateErrorType or None

error: Optional[*FirmwareUpdateErrorType*] = None

status: *FirmwareUpdateStatusType*

5.11 Firmware update status type

Firmware update statuses.

class wolk.model.firmware_update_status_type.**FirmwareUpdateStatusType**(*value*)

Bases: Enum

Enumeration of available firmware update status types.

ABORTED = 'ABORTED'

ERROR = 'ERROR'

INSTALLING = 'INSTALLING'

SUCCESS = 'SUCCESS'

5.12 Message

MQTT message model.

class wolk.model.message.**Message**(*topic: str*, *payload: Optional[Union[bytes, str]] = None*)

Bases: object

MQTT message identified by topic and payload.

Variables

- **topic** (*str*) – Topic where the message is from or will be sent to
- **payload** (*bytes or str or None*) – Content of the message

payload: Optional[Union[bytes, str]] = None

topic: str

5.13 Unit

Enumeration of available units on the Platform.

class wolk.model.unit.**Unit**(*value*)

Bases: Enum

Enumeration of default available units on the Platform.

Units are grouped by reading type, along with a comment that indicates what data type that unit expects.

```
AMPERE = 'AMPERE'

ANGSTROM = 'ANGSTROM'

ARE = 'ARE'

ASTRONOMICAL_UNIT = 'ASTRONOMICAL_UNIT'

ATMOSPHERE = 'ATMOSPHERE'

ATOM = 'ATOM'

ATOMIC_MASS = 'ATOMIC_MASS'

BAR = 'BAR'

BECQUEREL = 'BECQUEREL'

BIT = 'BIT'

BOOLEAN = 'BOOLEAN'

BYTE = 'BYTE'

C = 'C'

CANDELA = 'CANDELA'

CELSIUS = 'CELSIUS'

CENTIMETRE = 'CENTIMETRE'

CENTIRADIAN = 'CENTIRADIAN'

CENTIVOLT = 'CENTIVOLT'

COULOMB = 'COULOMB'

CUBIC_INCH = 'CUBIC_INCH'

CUBIC_METRE = 'CUBIC_METRE'

CURIE = 'CURIE'

DAY = 'DAY'

DAY_SIDEREAL = 'DAY_SIDEREAL'

DECIBEL = 'DECIBEL'

DECILITRE = 'DECILITRE'

DEGREE_ANGLE = 'DEGREE_ANGLE'

DYNE = 'DYNE'

E = 'E'

ELECTRON_MASS = 'ELECTRON_MASS'

ELECTRON_VOLT = 'ELECTRON_VOLT'
```

```
ERG = 'ERG'

FAHRENHEIT = 'FAHRENHEIT'

FARAD = 'FARAD'

FARADAY = 'FARADAY'

FOOT = 'FOOT'

FOOT_SURVEY_US = 'FOOT_SURVEY_US'

FRANKLIN = 'FRANKLIN'

G = 'G'

GALLON_DRY_US = 'GALLON_DRY_US'

GALLON_UK = 'GALLON_UK'

GAUSS = 'GAUSS'

GIGAHERTZ = 'GIGAHERTZ'

GILBERT = 'GILBERT'

GRADE = 'GRADE'

GRAM = 'GRAM'

GRAY = 'GRAY'

HECTARE = 'HECTARE'

HECTOPASCAL = 'HECTOPASCAL'

HENRY = 'HENRY'

HERTZ = 'HERTZ'

HORSEPOWER = 'HORSEPOWER'

HOUR = 'HOUR'

INCH = 'INCH'

INCH_OF_MERCURY = 'INCH_OF_MERCURY'

JOULE = 'JOULE'

KATAL = 'KATAL'

KELVIN = 'KELVIN'

KILOGRAM = 'KILOGRAM'

KILOGRAM_FORCE = 'KILOGRAM_FORCE'

KILOMETRE = 'KILOMETRE'

KILOMETRES_PER_HOUR = 'KILOMETRES_PER_HOUR'
```

```
KNOT = 'KNOT'

LAMBERT = 'LAMBERT'

LIGHT_YEAR = 'LIGHT_YEAR'

LITRE = 'LITRE'

LOCATION = 'LOCATION'

LUMEN = 'LUMEN'

LUX = 'LUX'

MACH = 'MACH'

MAXWELL = 'MAXWELL'

MEGAHERTZ = 'MEGAHERTZ'

METRE = 'METRE'

METRES_PER_SECOND = 'METRES_PER_SECOND'

METRES_PER_SQUARE_SECOND = 'METRES_PER_SQUARE_SECOND'

METRIC_TON = 'METRIC_TON'

MILE = 'MILE'

MILES_PER_HOUR = 'MILES_PER_HOUR'

MILLIBAR = 'MILLIBAR'

MILLILITRE = 'MILLILITRE'

MILLIMETER_OF_MERCURY = 'MILLIMETER_OF_MERCURY'

MILLIMETRE = 'MILLIMETRE'

MILLIVOLT = 'MILLIVOLT'

MINUTE = 'MINUTE'

MINUTE_ANGLE = 'MINUTE_ANGLE'

MOLE = 'MOLE'

MONTH = 'MONTH'

NAUTICAL_MILE = 'NAUTICAL_MILE'

NEWTON = 'NEWTON'

NUMERIC = 'NUMERIC'

OHM = 'OHM'

OUNCE = 'OUNCE'

OUNCE_LIQUID_UK = 'OUNCE_LIQUID_UK'
```

```
PARSEC = 'PARSEC'

PASCAL = 'PASCAL'

PERCENT = 'PERCENT'

PIXEL = 'PIXEL'

POINT = 'POINT'

POISE = 'POISE'

POUND = 'POUND'

POUND_FORCE = 'POUND_FORCE'

RAD = 'RAD'

RADIAN = 'RADIAN'

RANKINE = 'RANKINE'

REM = 'REM'

REVOLUTION = 'REVOLUTION'

ROENTGEN = 'ROENTGEN'

RUTHERFORD = 'RUTHERFORD'

SECOND = 'SECOND'

SECOND_ANGLE = 'SECOND_ANGLE'

SIEMENS = 'SIEMENS'

SIEVERT = 'SIEVERT'

SPHERE = 'SPHERE'

SQUARE_METRE = 'SQUARE_METRE'

STERADIAN = 'STERADIAN'

STOKE = 'STOKE'

TESLA = 'TESLA'

TEXT = 'TEXT'

TON_UK = 'TON_UK'

TON_US = 'TON_US'

VOLT = 'VOLT'

WATT = 'WATT'

WEBER = 'WEBER'

WEEK = 'WEEK'
```

```
YARD = 'YARD'

YEAR = 'YEAR'

YEAR_CALENDAR = 'YEAR_CALENDAR'

YEAR_SIDEREAL = 'YEAR_SIDEREAL'
```

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

W

INDEX

E

F

G

H

I

J

K

L

M

N

O

P

R

S

Y