Legend:
1. Pink boxes indicate a python file and the funciton that is executed there
2. Light blue boxes indicate a function being executed in the same file as the previous node of the tree
3. Green boxes indicate the file "mqtt_interaction_module.py"
4. Gray boxes indicate the file "mqtt_interaction_node_main.py"
5. Dark blue boxes indicate the file "mqtt_interaction_cloud.py"
6. Dashed boxes represent the "on_message" function events.
7. Dashes arrows explain a relationship between two functions, explain some rellevant function argument values, or indicate an event from/to the exterior.

WNI = WirepasNetworkInterface class inside "wirepas_network_interface.py" file

Written by Romà Masana on March 24th 2023

start_listen_nodes.py
MODE = LISTEN
INTERLOCUTOR = NODES
gw_backend_to_mesh_network_nodes.py (run_node_comunications(mode=2))
1. mqtt_interaction_node_main.py (init class)
mqtt_interaction_node_basic.py (init class)
mqtt_interaction_module.py (init class)
1. WNI (init class wni)
2. set_sink_and_gateway()
2. mqtt_interaction_module.py (listen)
mqtt_interaction_node_basic.py (listen_to_nodes())
1. WNI (register_data_cb())
Here we set cb = mqtt_interaction_module.on_message()
this enables the function on_message()
2. mqtt_interaction_module.py (start_listen_loop())
3. WNI (unregister_data_cb())
3. mqtt_interaction_module.py (on_message())
it triggers the function
when a node sends a message
mqtt_interaction_node_basic.py (on_message_nodes())
mqtt_interaction_node_main.py (on_message_main_node_listen())
interaction_tunnel.py (prepare_cloud_tunnel)
tunnel_to_cloud()
1. node_to_cloud_tunnel (init class)
mqtt_interaction_cloud.py (init class)
sets these variables for the new class
INTERLOCUTOR = CLOUD
MODE = LISTEN
start_listen_thread()
set_node_id()
set_sub_client()
1. paho mqtt "client.py" (init class Client)
2. on_subscribe()
3. paho mqtt "client.py" (username_pw_set())
4. mqtt_interaction_module.py (on_message())
topic = CLOUD_REQUEST_TO_NODETOPIC
triggers the function
when CLOUD sends a message
this enables the function on_message()
mqtt_interaction_cloud (on_message_c())
1. get_timestamp()
2. message_received_class.py (init class RxMsg)
3. message_received_class.py (msg.parse())
4. mqtt_interaction_module.py (update_with_new_msg())
5. message_received_class.py (msg.display())
6. interaction_tunnel.py (tunnel_to_node())
1. global_variables.py (get_supertopic())
2. cloud_to_node_tunnel (init class)
mqtt_interaction_node_temporal.py (init class)
mqtt_interaction_node_basic.py (init class)
mqtt_interaction.py (init class)
3. tunnel.transmit_message_to_node()
INTERLOCUTOR = NODES
MODE = REQUEST
PAYLOAD = None but ARGS given
mqtt_interac_node_basic.py (request_to_node())
1. global_variables.py (get_supertopic())
2. mqtt_interaction_module.py (convert_dst_address())
3. mqtt_interaction_module.py (update_msg_id())
4. create_message_functions (message_to_bytes())
5. global_variables.get_timestamp()
6. mqtt_interaction_module.py (publish())
1. global_variables.get_timestamp()
2. start_listen_thread()
1. listen()
mqtt_interation_node_basic.py (listen_to_nodes())
flow already described before
2. threading.py (start())
enables the on_message() function
3. mqtt_interaction_node_basic.py (publish_meshnet())
WNI (send_message())
send request to node
node
sends the response to the gateway
4. mqtt_interaction_module (on_message())
flow of what happens on a message response from a node, is similar to the node request to gateway.
7. update_gw_software.py (update_gw_software())
8. update_gw_database.py (update_gw_database())
9. read_gw_database.py (read_gw_database.py)
10. prepare_response.py (prepare_response())
11. mqtt_interaction_module.py (publish())
1. get_timestamp()
2. mqtt_interaction_cloud.py (publish_c())
1. get_timestamp()
2. paho mqtt "client.py" (publish())
response sent back to cloud
CLOUD
3. stop_sub_loop()
paho mqtt "client.py" (loop_stop())
5. connect_sub_client()
2. mqtt_interaction_module.py (start_listen_thread())
listen() (in a new thread)
mqtt_interaction_cloud (listen_to_cloud())
1. paho library client.py (loop_start())
2. mqtt_interaction_module (start_listen_loop())
threading.py (start())

Presented with xmind