**[midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan)/**[**custom\_components**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components)**/**[**midea\_ac\_lan**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan)**/**[**midea**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea)**/**[**devices**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/devices)**/**[**c3**](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/devices/c3)**/device.py**

|  |
| --- |
| Import logging |
|  | **Import logging – I assume ‘logging’ subroutine is located within the HA code.** from ...core.device import MiedaDevice Import class ‘MiedaDevice’ – from core.device Program located within [midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan)/[custom\_components](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components)/[midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan)/[midea](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea)/[core](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/core)/device.py |
|  | from .message import ( |
|  | MessageQuery, Import class ‘MessageQuery’ |
|  | MessageC3Response, Import class ‘MessageC3Response’ |
|  | MessageSet Import class ‘MessageSet' from [midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan)/[custom\_components](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components)/[midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan)/[midea](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea)/[devices](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/devices)/[c3](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/devices/c3)/message.py |
|  | ) |
|  | from ...backports.enum import StrEnum Import class ‘StrEnum’ from [midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan)/[custom\_components](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components)/[midea\_ac\_lan](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan)/[midea](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea)/[backports](https://github.com/georgezhao2010/midea_ac_lan/tree/master/custom_components/midea_ac_lan/midea/backports)/enum.py |
|  |  |
|  | \_LOGGER = logging.getLogger(\_\_name\_\_) |
|  |  |
|  |  |
|  | class DeviceAttributes(StrEnum): |
|  | zone1\_power = "zone1\_power" |
|  | zone2\_power = "zone2\_power" |
|  | dhw\_power = "dhw\_power" |
|  | zone1\_curve = "zone1\_curve" |
|  | zone2\_curve = "zone2\_curve" |
|  | disinfect = "disinfect" |
|  | fast\_dhw = "fast\_dhw" |
|  | zone\_temp\_type = "zone\_temp\_type" |
|  | zone1\_room\_temp\_mode = "zone1\_room\_temp\_mode" |
|  | zone2\_room\_temp\_mode = "zone2\_room\_temp\_mode" |
|  | zone1\_water\_temp\_mode = "zone1\_water\_temp\_mode" |
|  | zone2\_water\_temp\_mode = "zone2\_water\_temp\_mode" |
|  | mode = "mode" |
|  | mode\_auto = "mode\_auto" |
|  | zone\_target\_temp = "zone\_target\_temp" |
|  | dhw\_target\_temp = "dhw\_target\_temp" |
|  | room\_target\_temp = "room\_target\_temp" |
|  | zone\_heating\_temp\_max = "zone\_heating\_temp\_max" |
|  | zone\_heating\_temp\_min = "zone\_heating\_temp\_min" |
|  | zone\_cooling\_temp\_max = "zone\_cooling\_temp\_max" |
|  | zone\_cooling\_temp\_min = "zone\_cooling\_temp\_min" |
|  | tank\_actual\_temperature = "tank\_actual\_temperature" |
|  | room\_temp\_max = "room\_temp\_max" |
|  | room\_temp\_min = "room\_temp\_min" |
|  | dhw\_temp\_max = "dhw\_temp\_max" |
|  | dhw\_temp\_min = "dhw\_temp\_min" |
|  | target\_temperature = "target\_temperature" |
|  | temperature\_max = "temperature\_max" |
|  | temperature\_min = "temperature\_min" |
|  |  |
|  |  |
|  | class MideaC3Device(MiedaDevice): |
|  | def \_\_init\_\_( |
|  | self, |
|  | name: str, |
|  | device\_id: int, |
|  | ip\_address: str, |
|  | port: int, |
|  | token: str, |
|  | key: str, |
|  | protocol: int, |
|  | model: str, |
|  | customize: str |
|  | ): |
|  | super().\_\_init\_\_( |
|  | name=name, |
|  | device\_id=device\_id, |
|  | device\_type=0xC3, |
|  | ip\_address=ip\_address, |
|  | port=port, |
|  | token=token, |
|  | key=key, |
|  | protocol=protocol, |
|  | model=model |
|  | ) |
|  | self.\_attributes = { |
|  | DeviceAttributes.zone1\_power: False, |
|  | DeviceAttributes.zone2\_power: False, |
|  | DeviceAttributes.dhw\_power: False, |
|  | DeviceAttributes.zone1\_curve: False, |
|  | DeviceAttributes.zone2\_curve: False, |
|  | DeviceAttributes.disinfect: False, |
|  | DeviceAttributes.fast\_dhw: False, |
|  | DeviceAttributes.zone\_temp\_type: [False, False], |
|  | DeviceAttributes.zone1\_room\_temp\_mode: False, |
|  | DeviceAttributes.zone2\_room\_temp\_mode: False, |
|  | DeviceAttributes.zone1\_water\_temp\_mode: False, |
|  | DeviceAttributes.zone2\_water\_temp\_mode: False, |
|  | DeviceAttributes.mode: 1, |
|  | DeviceAttributes.mode\_auto: 1, |
|  | DeviceAttributes.zone\_target\_temp: [25, 25], |
|  | DeviceAttributes.dhw\_target\_temp: 25, |
|  | DeviceAttributes.room\_target\_temp: 30, |
|  | DeviceAttributes.zone\_heating\_temp\_max: [55, 55], |
|  | DeviceAttributes.zone\_heating\_temp\_min: [25, 25], |
|  | DeviceAttributes.zone\_cooling\_temp\_max: [25, 25], |
|  | DeviceAttributes.zone\_cooling\_temp\_min: [5, 5], |
|  | DeviceAttributes.room\_temp\_max: 60, |
|  | DeviceAttributes.room\_temp\_min: 34, |
|  | DeviceAttributes.dhw\_temp\_max: 60, |
|  | DeviceAttributes.dhw\_temp\_min: 20, |
|  | DeviceAttributes.tank\_actual\_temperature: None, |
|  | DeviceAttributes.target\_temperature: [25, 25], |
|  | DeviceAttributes.temperature\_max: [0, 0], |
|  | DeviceAttributes.temperature\_min: [0, 0] |
|  | } |
|  |  |
|  | def build\_query(self): |
|  | return [MessageQuery(self.\_device\_protocol\_version)] |
|  |  |
|  | def process\_message(self, msg): |
|  | message = MessageC3Response(msg) |
|  | \_LOGGER.debug(f"[{self.device\_id}] Received: {message}") |
|  | new\_status = {} |
|  | for status in self.\_attributes.keys(): |
|  | if hasattr(message, status.value): |
|  | self.\_attributes[status] = getattr(message, status.value) |
|  | new\_status[status.value] = getattr(message, status.value) |
|  | if len(new\_status) > 0: |
|  | for zone in [0, 1]: |
|  | if self.\_attributes[DeviceAttributes.zone\_temp\_type][zone]: # Water temp mode |
|  | self.\_attributes[DeviceAttributes.target\_temperature][zone] = \ |
|  | self.\_attributes[DeviceAttributes.zone\_target\_temp][zone] |
|  | if self.\_attributes[DeviceAttributes.mode\_auto] == 2: # cooling mode |
|  | self.\_attributes[DeviceAttributes.temperature\_max][zone] = \ |
|  | self.\_attributes[DeviceAttributes.zone\_cooling\_temp\_max][zone] |
|  | self.\_attributes[DeviceAttributes.temperature\_min][zone] = \ |
|  | self.\_attributes[DeviceAttributes.zone\_cooling\_temp\_min][zone] |
|  | elif self.\_attributes[DeviceAttributes.mode] == 3: # heating mode |
|  | self.\_attributes[DeviceAttributes.temperature\_max][zone] = \ |
|  | self.\_attributes[DeviceAttributes.zone\_heating\_temp\_max][zone] |
|  | self.\_attributes[DeviceAttributes.temperature\_min][zone] = \ |
|  | self.\_attributes[DeviceAttributes.zone\_heating\_temp\_min][zone] |
|  | else: # Room temp mode |
|  | self.\_attributes[DeviceAttributes.target\_temperature][zone] = \ |
|  | self.\_attributes[DeviceAttributes.room\_target\_temp] |
|  | self.\_attributes[DeviceAttributes.temperature\_max][zone] = \ |
|  | self.\_attributes[DeviceAttributes.room\_temp\_max] |
|  | self.\_attributes[DeviceAttributes.temperature\_min][zone] = \ |
|  | self.\_attributes[DeviceAttributes.room\_temp\_min] |
|  | if self.\_attributes[DeviceAttributes.zone1\_power]: |
|  | if self.\_attributes[DeviceAttributes.zone\_temp\_type][zone]: |
|  | self.\_attributes[DeviceAttributes.zone1\_water\_temp\_mode] = True |
|  | self.\_attributes[DeviceAttributes.zone1\_room\_temp\_mode] = False |
|  | else: |
|  | self.\_attributes[DeviceAttributes.zone1\_water\_temp\_mode] = False |
|  | self.\_attributes[DeviceAttributes.zone1\_room\_temp\_mode] = True |
|  | else: |
|  | self.\_attributes[DeviceAttributes.zone1\_water\_temp\_mode] = False |
|  | self.\_attributes[DeviceAttributes.zone1\_room\_temp\_mode] = False |
|  | if self.\_attributes[DeviceAttributes.zone2\_power]: |
|  | if self.\_attributes[DeviceAttributes.zone\_temp\_type][zone]: |
|  | self.\_attributes[DeviceAttributes.zone2\_water\_temp\_mode] = True |
|  | self.\_attributes[DeviceAttributes.zone2\_room\_temp\_mode] = False |
|  | else: |
|  | self.\_attributes[DeviceAttributes.zone2\_water\_temp\_mode] = False |
|  | self.\_attributes[DeviceAttributes.zone2\_room\_temp\_mode] = True |
|  | else: |
|  | self.\_attributes[DeviceAttributes.zone2\_water\_temp\_mode] = False |
|  | self.\_attributes[DeviceAttributes.zone2\_room\_temp\_mode] = False |
|  | new\_status[DeviceAttributes.zone1\_water\_temp\_mode.value] = \ |
|  | self.\_attributes[DeviceAttributes.zone1\_water\_temp\_mode] |
|  | new\_status[DeviceAttributes.zone2\_water\_temp\_mode.value] = \ |
|  | self.\_attributes[DeviceAttributes.zone2\_water\_temp\_mode] |
|  | new\_status[DeviceAttributes.zone1\_room\_temp\_mode.value] = \ |
|  | self.\_attributes[DeviceAttributes.zone1\_room\_temp\_mode] |
|  | new\_status[DeviceAttributes.zone2\_room\_temp\_mode.value] = \ |
|  | self.\_attributes[DeviceAttributes.zone2\_room\_temp\_mode] |
|  |  |
|  | return new\_status |
|  |  |
|  | def make\_message\_set(self): |
|  | message = MessageSet(self.\_device\_protocol\_version) |
|  | message.zone1\_power = self.\_attributes[DeviceAttributes.zone1\_power] |
|  | message.zone2\_power = self.\_attributes[DeviceAttributes.zone2\_power] |
|  | message.dhw\_power = self.\_attributes[DeviceAttributes.dhw\_power] |
|  | message.mode = self.\_attributes[DeviceAttributes.mode] |
|  | message.zone\_target\_temp = self.\_attributes[DeviceAttributes.zone\_target\_temp] |
|  | message.dhw\_target\_temp = self.\_attributes[DeviceAttributes.dhw\_target\_temp] |
|  | message.room\_target\_temp = self.\_attributes[DeviceAttributes.room\_target\_temp] |
|  | message.zone1\_curve = self.\_attributes[DeviceAttributes.zone1\_curve] |
|  | message.zone2\_curve = self.\_attributes[DeviceAttributes.zone2\_curve] |
|  | message.disinfect = self.\_attributes[DeviceAttributes.disinfect] |
|  | message.fast\_dhw = self.\_attributes[DeviceAttributes.fast\_dhw] |
|  | return message |
|  |  |
|  | def set\_attribute(self, attr, value): |
|  | if attr in [ |
|  | DeviceAttributes.zone1\_power, |
|  | DeviceAttributes.zone2\_power, |
|  | DeviceAttributes.dhw\_power, |
|  | DeviceAttributes.zone1\_curve, |
|  | DeviceAttributes.zone2\_curve, |
|  | DeviceAttributes.disinfect, |
|  | DeviceAttributes.fast\_dhw, |
|  | DeviceAttributes.dhw\_target\_temp |
|  | ]: |
|  | message = self.make\_message\_set() |
|  | setattr(message, str(attr), value) |
|  | self.build\_send(message) |
|  |  |
|  | def set\_mode(self, zone, mode): |
|  | message = self.make\_message\_set() |
|  | if zone == 0: |
|  | message.zone1\_power = True |
|  | else: |
|  | message.zone2\_power = True |
|  | message.mode = mode |
|  | self.build\_send(message) |
|  |  |
|  | def set\_target\_temperature(self, zone, target\_temperature, mode): |
|  | message = self.make\_message\_set() |
|  | if self.\_attributes[DeviceAttributes.zone\_temp\_type][zone]: |
|  | message.zone\_target\_temp[zone] = target\_temperature |
|  | else: |
|  | message.room\_target\_temp = target\_temperature |
|  | if mode is not None: |
|  | if zone == 0: |
|  | message.zone1\_power = True |
|  | else: |
|  | message.zone2\_power = True |
|  | message.mode = mode |
|  | self.build\_send(message) |
|  |  |
|  | @property |
|  | def attributes(self): |
|  | return super().attributes |
|  |  |
|  |  |
|  | class MideaAppliance(MideaC3Device): |
|  | pass |