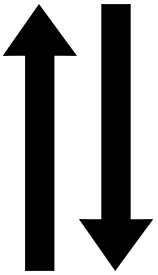
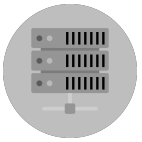
Paternoster ERP - Eleon S1

Implementation und Dokumentation des eleon S1 Universal Monitoring Gateway Schnittstelle bzw. API.

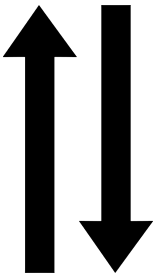
Version 1.0

eleonS1 MQTT Cloud





basic-erp MQTT Server



Paternoster ERP Instanz

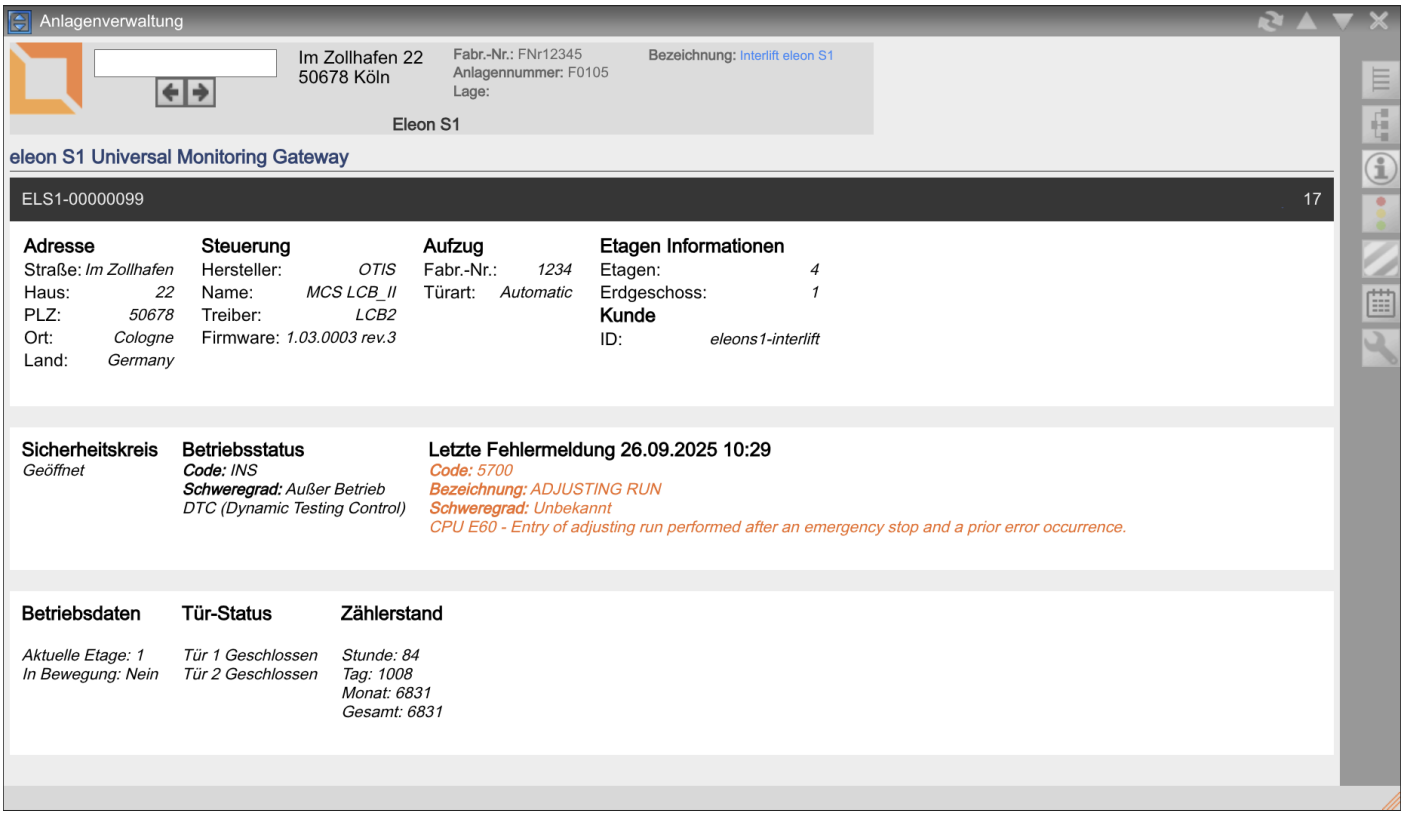
Paternoster ERP Instanz

Paternoster ERP Instanz

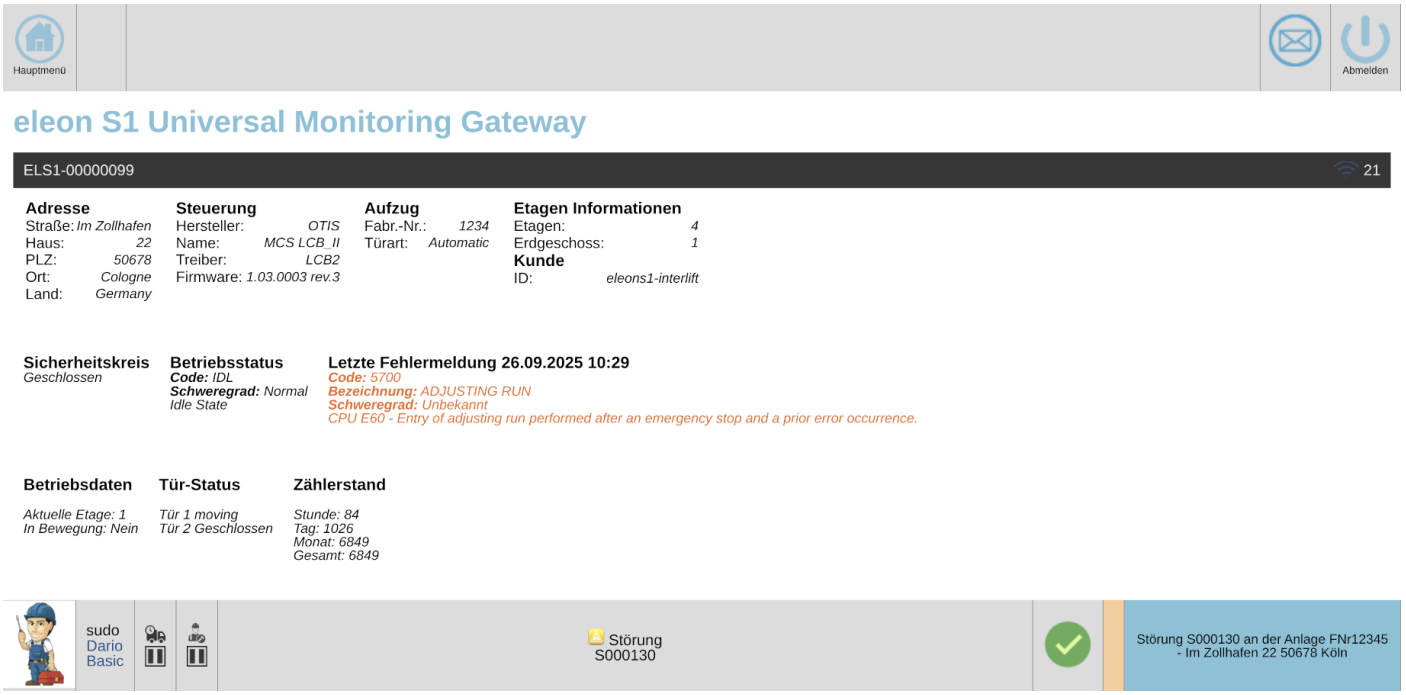
Paternoster ERP Instanz

Arbeitsablauf

Hauptprogramm

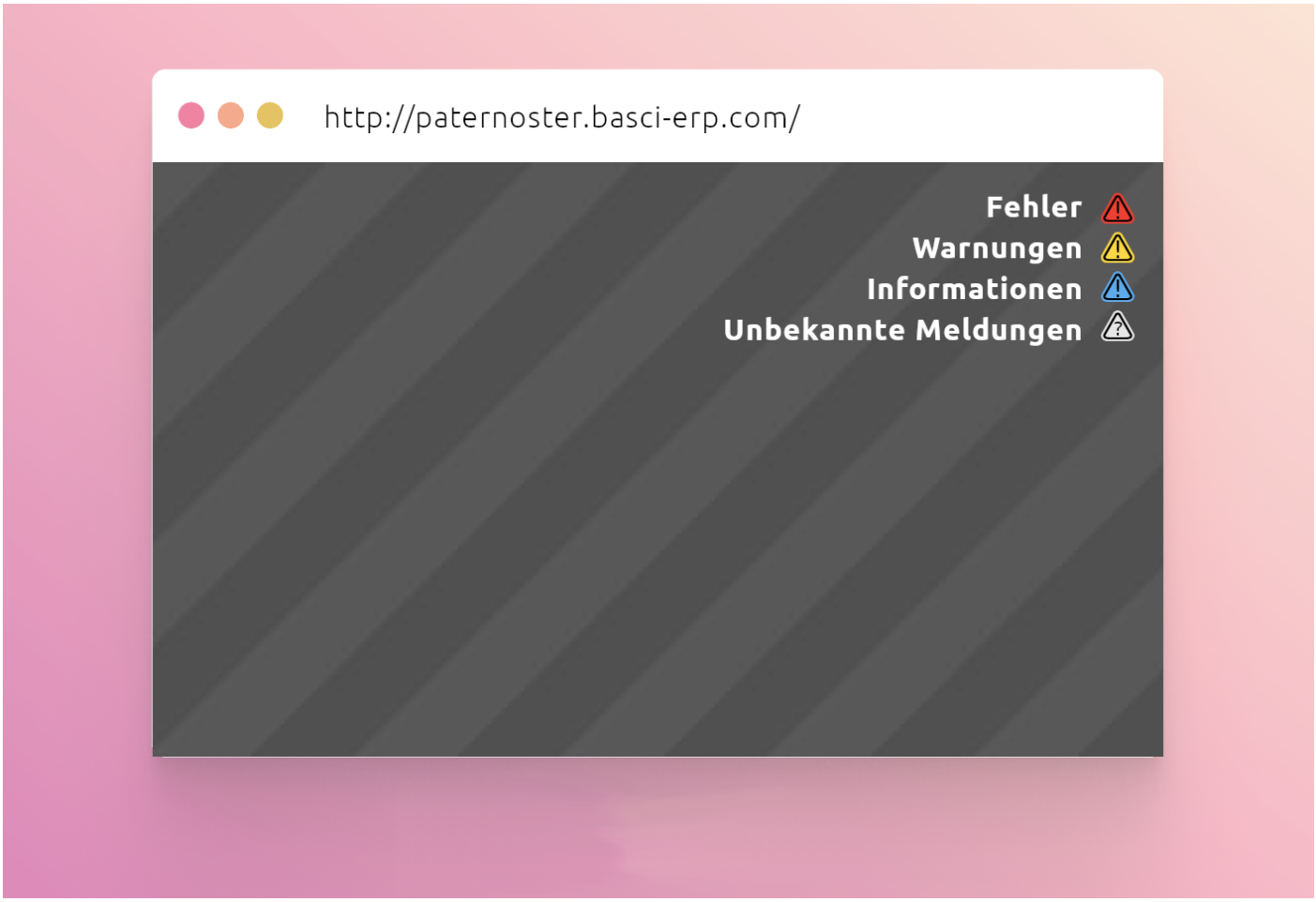


Mobile App



Die Informationen der Steuergeräte werden im Hauptprogramm in der Anlagenverwaltung bei den jeweiligen Anlagen abgebildet. In der mobile App gibt es dazu einen eigenen Bereich.

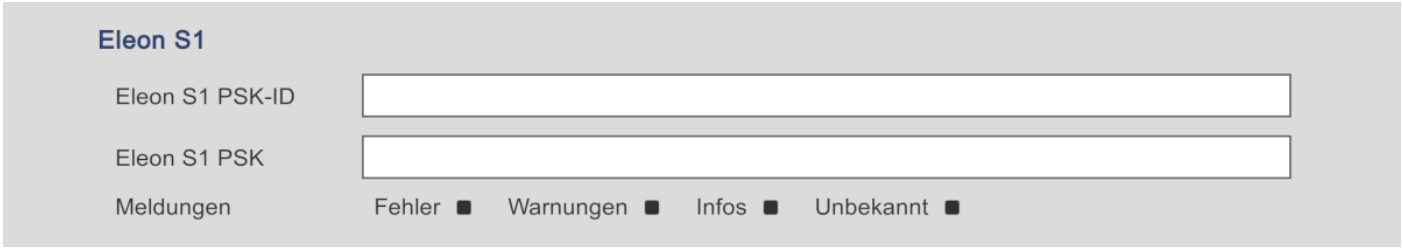
Da nicht davon auszugehen ist, das eine Anlage, bei der gerade ein Störfall auftritt auch vom Benutzer ausgewählt ist, werden alle Meldungen der Steuergeräte durch unabhängige Prozesse überwacht und den Benutzern mitgeteilt. Dabei unterscheidet Paternoster ERP zwischen vier Meldearten. Fehler, Warnungen, Informationen und Unbekannt. Diese Meldungen werden bei den Benutzern am Desktop des Hauptprogramms dargestellt.



Die Meldearten bzw. deren Symbole werden nur einmal abgebildet, die betroffene(n) Anlage(n) werden zu den jeweiligen Symbolen aufgelistet und können durch einen Mausklick aufgerufen werden.

Einstellungen

Laufzeiteinstellungen für die Kommunikationen zwischen den Paternoster ERP Instanzen und dem basic-erp MQTT-Server werden im Konfigurationsbereich der Anlagenverwaltung vorgenommen.



- **Eleon S1 PSK-ID**: Die MQTT Kunden ID.
- **Eleon S1 PSK**: TLS-PSK-Verschlüsselung, „Preshared Key“.
- **Meldungen**: Die Meldearten, über die die Benutzer informiert werden sollen.

Entwickler

API Module und Aufbau.

- **EleonS1Constants**:
  Diese Bibliothek enthält alle Schlüssel zur Kommunikation zwischen der Paternoster ERP Instanz und dem basic-erp MQTT-Server.
- **EleonS1Srv**: Dieser Dienst baut die Verbindung zwischen der Paternoster ERP Instanz und dem basic-erp MQTT-Server auf und stellt die Anfragen an den basic-erp MQTT-Server.
- **EleonS1SrvReader**: Dieser Dienst wertet die vom basic-erp MQTT-Server gesendeten Daten aus.

EleonS1Constants

```
import ( 'lib', 'eleons1_constants' ) ;
```

/** Current date */
API_DATE
/** Current time HH:ii:ss */
API_TIME
/** Current unix time */
API_UNIX
/** Status as boolean */
API_STATUS
/** Possible error code */
API_ERROR_CODE
/** Possible error message */
API_ERROR_MESSAGE
/** Possible message */
API_MESSAGE
/** Request licence key */
API_LICENCE
/** Request gateway key */
API_GATEWAY
/** Request controller key */
API_CONTROLLER_KEY
/** Request controller Eleon S1 */
API_CONTROLLER_ELEONS1
/** Request action key */
API_ACTION_KEY
/** Request action for prepared gateway data */
API_ACTION_PREPARED
/** Request action for fetch gateway data */
API_ACTION_FETCH
/** Request action for gateway list */
API_ACTION_LIST
/** Request PSK key */
API_PSK
/** Request PSK identity key */
API_PSK_ID
/** Data key of request container */
API_DATA
/** Data key for timestamp from device to MQTT broker */
API_KEY_TIME
/** Data key for value from device to MQTT broker */
API_KEY_VALUE
/** Data key for meta data */
API_KEY_META
/** Data key for meta manufacturer */
API_KEY_META_MANUFACTURER

/** Data key for meta controller */
API_KEY_META_CONTROLLER
/** Data key for meta driver */
API_KEY_META_DRIVER
/** Data key for meta floor info */
API_KEY_META_FLOOR
/** Data key for meta floor ground */
API_KEY_META_FLOOR_GROUND
/** Data key for meta floor count */
API_KEY_META_FLOOR_COUNT
/** Data key for meta factory number */
API_KEY_META_FACTORY_NUMBER
/** Data key for meta address */
API_KEY_META_ADDRESS
/** Data key for meta address country */
API_KEY_META_ADDRESS_COUNTRY
/** Data key for meta address postcode */
API_KEY_META_ADDRESS_POSTCODE
/** Data key for meta address place */
API_KEY_META_ADDRESS_PLACE
/** Data key for meta address street */
API_KEY_META_ADDRESS_STREET
/** Data key for meta address house */
API_KEY_META_ADDRESS_HOUSE
/** Data key for meta door type */
API_KEY_META_DOOR_TYPE
/** Data key for meta firmware */
API_KEY_META_FIRMWARE
/** Data key for device status */
API_KEY_DEVICE_STATUS
/** Data key for controller connection */
API_KEY_CONNECTED_TO_CONTROLLER
/** Data key for strenth of device singnal */
API_KEY_SIGNAL_STRENGTH
/** Data key for last transmission from device to MQTT broker */
API_KEY_LAST_TRANSMISSION
/** Data key for gateway customer */
API_KEY_CUSTOMER
/** Data key for gateway customer key */
API_KEY_CUSTOMER_KEY
/** Data key for gateway elevator information current floor relative key */
API_KEY_ELEVATOR_INFO_CUR_FLOOR_REL
/** Data key for gateway elevator information current floor absolute key */
API_KEY_ELEVATOR_INFO_CUR_FLOOR_ABS
/** Data key for gateway elevator information moving key */
API_KEY_ELEVATOR_INFO_MOVING
/** Data key for gateway elevator information door */
API_KEY_ELEVATOR_INFO_DOOR
/** Data key for gateway elevator information door state */
API_KEY_ELEVATOR_INFO_DOOR_STATE

/** Data key for gateway elevator KPI floor stops relative */
API_KEY_ELEVATOR_KPI_FLOOR_STOPS_REL
/** Data key for gateway elevator KPI floor stops absolute */
API_KEY_ELEVATOR_KPI_FLOOR_STOPS_ABS
/** Data key for safety chain closed */
API_KEY_ELEVATOR_INFO_SAFETY_CHAIN_CLOSED
/** Data key for elevator KPI ride counter total */
API_KEY_ELEVATOR_KPI_RIDE_COUNTER_TOT
/** Data key for elevator KPI ride counter hour */
API_KEY_ELEVATOR_KPI_RIDE_COUNTER_HOU
/** Data key for elevator KPI ride counter day */
API_KEY_ELEVATOR_KPI_RIDE_COUNTER_DAY
/** Data key for elevator KPI ride counter month */
API_KEY_ELEVATOR_KPI_RIDE_COUNTER_MON
/** Data key for alarm error */
API_KEY_ALARM_ERROR
/** Data key for alarm error time */
API_KEY_ALARM_ERROR_TIME
/** Data key for alarm error code */
API_KEY_ALARM_ERROR_CODE
/** Data key for alarm error name */
API_KEY_ALARM_ERROR_NAME
/** Data key for alarm error description */
API_KEY_ALARM_ERROR_DESC
/** Data key for alarm error severity */
API_KEY_ALARM_ERROR_SEVERITY
/** Data key for alarm error mode */
API_KEY_ALARM_ERROR_MODE
/** Data key for alarm error I/O value */
API_KEY_ALARM_ERROR_IO
/** Data key for alarm status */
API_KEY_ALARM_STATUS
/** Data key for alarm status code */
API_KEY_ALARM_STATUS_CODE
/** Data key for alarm status description */
API_KEY_ALARM_STATUS_DESC
/** Data key for alarm status severity */
API_KEY_ALARM_STATUS_SEVERITY
/** Data key for alarm status mode */
API_KEY_ALARM_STATUS_MODE
/** Data key for alarm status I/O value */
API_KEY_ALARM_STATUS_IO
/** Data key for gateway */
API_KEY_GATEWAY
/** Name of the ping file */
API_PING
/** Name of the connect file */
API_CONNECT
/** Name of the index file */
API_INDEX

EleonS1Srv

```
import ( 'srv', 'eleons1_srv' ) ;
```

```
/**
* Eleon S1 service for data query from the MQTT server
*
* @param string $psk pre shared key
* @param string $pid PSK-Identity or customer ID
*
* @return void
*/

public function __construct ( $psk, $pid ) ;

/**
* To transform array into a JSON string
*
* @param array $dat Data container
* @param int $flg Optional flags
* @param int $dep Optional depth
*
* @return string
*/

public static function toJson ( $dat, $flg = null, $dep = null ) ;

/**
* To transform JSON into array or object
*
* @param string $jso JSON
* @param bool $aso true = associative arrays; false = object, null = associative array or object
depending
JSON_OBJECT_AS_ARRAY is flags.
* @param int $dep Optional depth
* @param int $flg Optional flags
*
* @return array Datacontainer from JSON string
*/

public static function fromJson ( $jso, $aso = null, $dep = null, $flg = null ) ;
```

```
/**
* Requested Eleon S1 gateway data as HTML
*
* @param string $jso JSON Gateway data
* @param string $she Gateway
* @param string $wrp Tag wrapper
*
* @return string HTML
*/

public function html ( $jso, $she, $wrp = '%' ) ;

/**
* Ping test to MQTT server
*
* @return string JSON
*/

public function ping () ;

/**
* Connection test to MQTT server
*
* @return string JSON
*/

public function connect () ;

/**
* All device errors of requested list data
*
* @param string $jso raw, unprocessed JSON from MQTT Server
*
* @return array container with EleonS1SrvReader objects or empty array
*/

public function errors ( $jso ) ;

/**
* All device warnings of requested list data
*
* @param string $jso raw, unprocessed JSON from MQTT Server
*
* @return array container with EleonS1SrvReader objects or empty array
*/

public function warnings ( $jso ) ;
```

```
/**
* All device infos of requested list data
*
* @param string $jso raw, unprocessed JSON from MQTT Server
*
* @return array container with EleonS1SrvReader objects or empty array
*/

public function infos ( $jso ) ;

/**
* All device unknown errors of requested list data
*
* @param string $jso raw, unprocessed JSON from MQTT Server
*
* @return array container with EleonS1SrvReader objects or empty array
*/

public function unknown ( $jso ) ;

/**
* All device history data
*
* @param string $gat Gateway
*
* @return array container with history data
*/

public function history ( $gat ) ;

/**
* To set History Data
*
* @param EleonS1SrvReader $objRdr Gateway Reader Object
*
* @return bool
*/

public function setHistory ( EleonS1SrvReader $objRdr ) ;

/**
* JSON data of a gateway
*
* @param string $gat Gateway
*
* @return string JSON
*/

public function getGat ( $gat ) ;

/**
* JSON data of all gateways
*
* @return string JSON
*/
```

public function getLst () ;

```
/**
* Default request data
*
* @param bool $jso As JSON instead of array
* @param string $cod Error code
* @param string $err Error text
* @param string $msg Message text
*
* @return mixed JSON/Array
*/
```

public function getDrd ($jso = false, $cod = null, $err = null, $msg = null) ;

EleonS1SrvReader

```
import ( 'srv', 'eleons1_reader' ) ;
```

```
/**
* Eleon S1 service to read query from the MQTT server
* NOTE: pass second argument to hand over a valid gateway data container,
* which is preferred to the first argument
*
* @param string $jso JSON Gateway data
* @param array $dat Optional direct the gateway data container
*
* @return void
*/

public function __construct ( $jso, $dat = null ) ;

/**
* To transform unix time in system date format
*
* @param long $tim Unix time
* @param string $tip Time pattern
*
* @return string
*/

public static function toDate ( $tim, $tip = ' H:i:s' ) ;

/**
* To transform array into a JSON string
*
* @param array $dat Data container
* @param int $flg Optional flags
* @param int $dep Optional depth
*
* @return string
*/

public static function toJson ( $dat, $flg = null, $dep = null ) ;
```

```
/**
* To transform JSON into array or object
*
* @param string $jso JSON
* @param bool $aso true = associative arrays; false = object, null = associative array or object
depending JSON_OBJECT_AS_ARRAY is flags.
* @param int $dep Optional depth
* @param int $flg Optional flags
*
* @return array Datacontainer from JSON string
*/

public static function fromJson ( $jso, $aso = null, $dep = null, $flg = null ) ;

/**
* Whether the controller is in idle or normal state
*
* @return bool
*/

public function isIdle () ;

/**
* To get the state of the safety chain
* Use getSafetyChainStatus to get a boolen value
*
* @return string
*/

public function isSafetyChainClosed () ;

/**
* Whether an alarm error has occurred
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isError ( $int = false ) ;

/**
* Whether an alarm error is unknown
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer */

public function isErrorUnknown ( $int = false ) ;
```

```
/**
* Whether an alarm error is information
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isErrorInfo ( $int = false ) ;

/**
* Whether an alarm error is a warning
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isErrorWarning ( $int = false ) ;

/**
* Whether an alarm error is a failure
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isErrorFailure ( $int = false ) ;

/**
* Whether an alarm status has occurred
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isStatus ( $int = false ) ;

/**
* Whether an alarm status is unknown
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isStatusUnknown ( $int = false ) ;
```

```
/**
* Whether an alarm status is normal
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isStatusNormal ( $int = false ) ;

/**
* Whether an alarm status is special
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isStatusSpecial ( $int = false ) ;

/**
* Whether an alarm status is inspection
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isStatusInspection ( $int = false ) ;

/**
* Whether an alarm status is out of order
*
* @param bool $int Return as integer
*
* @return mixed Boolean or Integer
*/

public function isStatusOutOfOrder ( $int = false ) ;

/**
* To get complete requested MQTT JSON as data container
*
* @return array
*/

public function getReq () ;
```

```
/**
* To get complete data section from requ+ested MQTT data container
*
* @return array
*/

public function getDat () ;

/**
* To get value from given data container
*
* @param array $dat Data container
* @param string $key Data key
* @param mixed $ret Default return value
* @param bool $boo Boolean value expected
*
* @return mixed
*/

public function getVal ( $dat, $key, $ret = '', $boo = false ) ;

/**
* To get meta section of data section from requested MQTT data
*
* @return array
*/

public function getMet () ;

/**
* To get floor info section of data section from requested MQTT data
*
* @return array
*/

public function getFlo () ;

/**
* To get address section of data section from requested MQTT data
*
* @return array
*/

public function getAdr () ;
```

```
/**
* To get time from given data container
* NOTE: data container must contain API time key
*
* @param array $dat Data container
* @param string $key Data key
* @param bool $uni Return as long or Unix time
* @param string $tip Time pattern
*
* @return array
*/

public function getTim ( $dat, $uni = false, $tip = ' H:i' ) ;

/**
* To get alarm error and status severity code/value data container as associative array
* Note: The index is the corresponding code or integer value
*
* @return array
*/

public function getAsv () ;

/**
* To get the MQTT gateway
*
* @return string
*/

public function getGateway () ;

/**
* To get the elevator manufacturer
*
* @return string
*/

public function getManufacturer () ;

/**
* To get the elevator serial or factory number
*
* @return string
*/

public function getFactoryNumber () ;
```

```
/**
* To get the elevator door type
*
* @return string
*/

public function getDoorType () ;

/**
* To get data number of doors transmitted by MQTT data
*
* @return int
*/

public function getDoorCount () ;

/**
* To get data container of all doors
*
* @return array
*/

public function getDoorInfoData () ;

/**
* To get the last door status time
*
* @param int $flo Floor
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getDoorStatusTime ( $flo, $uni = false ) ;

/**
* To get the door status
*
* @param int $flo Floor
*
* @return string
*/

public function getDoorStatusValue ( $flo ) ;

/**
* To get number of relative (0-X) floors counted
*
* @return int
*/

public function getFloorStopsCountRel () ;
```

```
/**
* To get number of absolute (1-X) floors counted
*
* @return int
*/

public function getFloorStopsCountAbs () ;

/**
* To get data container of all relative (0-X) floor stops
*
* @return array
*/

public function getFloorStopsRelInfoData () ;

/**
* To get data container of all absolute (1-X) floor stops
*
* @return array
*/

public function getFloorStopsAbsInfoData () ;

/**
* To get the last relative (0-X) door stop time
*
* @param int $flo Floor
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getFloorStopRelTime ( $flo, $uni = false ) ;

/**
* To get the last absolute (1-X) door stop time
*
* @param int $flo Floor
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getFloorStopAbsTime ( $flo, $uni = false ) ;
```

```
/**
* To get the stops of relative (0-X) floor
*
* @param int $flo Floor
*
* @return int
*/

public function getFloorStopRelValue ( $flo ) ;

/**
* To get the stops of absolute (1-X) floor
*
* @param int $flo Floor
*
* @return int
*/

public function getFloorStopAbsValue ( $flo ) ;

/**
* To get counter last reading time for hourly trips
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getTravelCounterHouTime ( $uni = false ) ;

/**
* To get counter last reading time for daily trips
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getTravelCounterDayTime ( $uni = false ) ;

/**
* To get counter last reading time for monthly trips
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getTravelCounterMonTime ( $uni = false ) ;
```

```
/**
* To get counter last reading time for total trips
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getTravelCounterTotTime ( $uni = false ) ;

/**
* To get counter for hourly trips
*
* @return int
*/

public function getTravelCounterHouValue () ;

/**
* To get counter for daily trips
*
* @return int
*/

public function getTravelCounterDayValue () ;

/**
* To get counter for monthly trips
*
* @return int
*/

public function getTravelCounterMonValue () ;

/**
* To get counter for total trips
*
* @return int
*/

public function getTravelCounterTotValue () ;

/**
* To get the device controller
*
* @return string
*/

public function getController () ;
```

```
/**
* To get the device driver
*
* @return string
*/

public function getDriver () ;

/**
* To get the device firmware
*
* @return string
*/

public function getFirmware () ;

/**
* To get number of ground floors
*
* @return int
*/

public function getGroundFloor () ;

/**
* To get number of floors
*
* @return int
*/

public function getFloorCount () ;

/**
* To get the street in which the elevator is located
*
* @return string
*/

public function getStreet () ;

/**
* To get the street house number in which the elevator is located
*
* @return string
*/

public function getHouse () ;
```

```
/**
* To get the postcode of the elevator location
*
* @return string
*/

public function getPostcode () ;

/**
* To get the place of the elevator location
*
* @return string
*/

public function getPlace () ;

/**
* To get the country in which the elevator is located
*
* @return string
*/

public function getCountry () ;

/**
* To get the customer name or PSK-Identity
*
* @return string
*/

public function getCustomer () ;

/**
* To get the device last transmission
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getLastTransmissionTime ( $uni = false ) ;

/**
* To get the device last transmission value
*
* @param bool $uni Return as long or Unix time
*
* @return mixed
*/

public function getLastTransmissionValue ( $uni = false ) ;
```

```
/**
* To get the device last status time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getDeviceStatusTime ( $uni = false ) ;

/**
* To get the device last status value
*
* @return int
*/

public function getDeviceStatusValue () ;

/**
* To get the device last controller connection time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getControllerConnTime ( $uni = false ) ;

/**
* To get the device controller connection value
*
* @return int
*/

public function getControllerConnValue () ;

/**
* To get the device last signal strength time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getSignalStrengthTime ( $uni = false ) ;

/**
* To get the device signal strength value
*
* @return int
*/

public function getSignalStrengthValue () ;
```

```
/**
* To get the last moving time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getMovingTime ( $uni = false ) ;

/**
* To get the moving status
*
* @return bool
*/

public function getMovingStatus () ;

/**
* To get the moving status as text
*
* @return string Yes/No
*/

public function getMovingStatusText () ;

/**
* To get the last relative (0-X) current floor time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getCurrentFloorRelTime ( $uni = false ) ;

/**
* To get the current relative (0-X) floor
*
* @return int
*/

public function getCurrentFloorRel () ;

/**
* To get the last absolute (1-X) current floor time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getCurrentFloorAbsTime ( $uni = false ) ;
```

```
/**
* To get the current absolute (1-X) floor
*
* @return int
*/

public function getCurrentFloorAbs () ;

/**
* To get the last safety chain time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getSafetyChainTime ( $uni = false ) ;

/**
* To get the safety chain state
*
* @return bool
*/

public function getSafetyChainStatus () ;

/**
* To get the whole alarm error data container
*
* @return array
*/

public function getErrorData () ;

/**
* To get the alarm error time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getErrorTime ( $uni = false ) ;

/**
* To get the alarm error code
*
* @return int
*/

public function getErrorCode () ;
```

```
/**
* To get the alarm error name
*
* @return string
*/

public function getErrorName () ;

/**
* To get the alarm error description
*
* @return string
*/

public function getErrorDesc () ;

/**
* To get the alarm error mode
*
* @return string
*/

public function getErrorMode () ;

/**
* To get the alarm error severity
*
* @return int
*/

public function getErrorSeverity () ;

/**
* To get the alarm error severity as text
*
* @return string
*/

public function getErrorSeverityText () ;

/**
* To get the whole alarm status data container
*
* @return array
*/

public function getStatusData () ;
```

```
/**
* To get the alarm status time
*
* @param bool $uni Return as long or Unix time
*
* @return mixed Long or system date format
*/

public function getStatusTime ( $uni = false ) ;

/**
* To get the alarm status code
*
* @return string
*/

public function getStatusCode () ;

/**
* To get the alarm status description
*
* @return string
*/

public function getStatusDesc () ;

/**
* To get the alarm status mode
*
* @return string
*/

public function getStatusMode () ;

/**
* To get the alarm status severity
*
* @return int
*/

public function getStatusSeverity () ;

/**
* To get the alarm status severity as text
*
* @return string
*/

public function getStatusSeverityText () ;
```