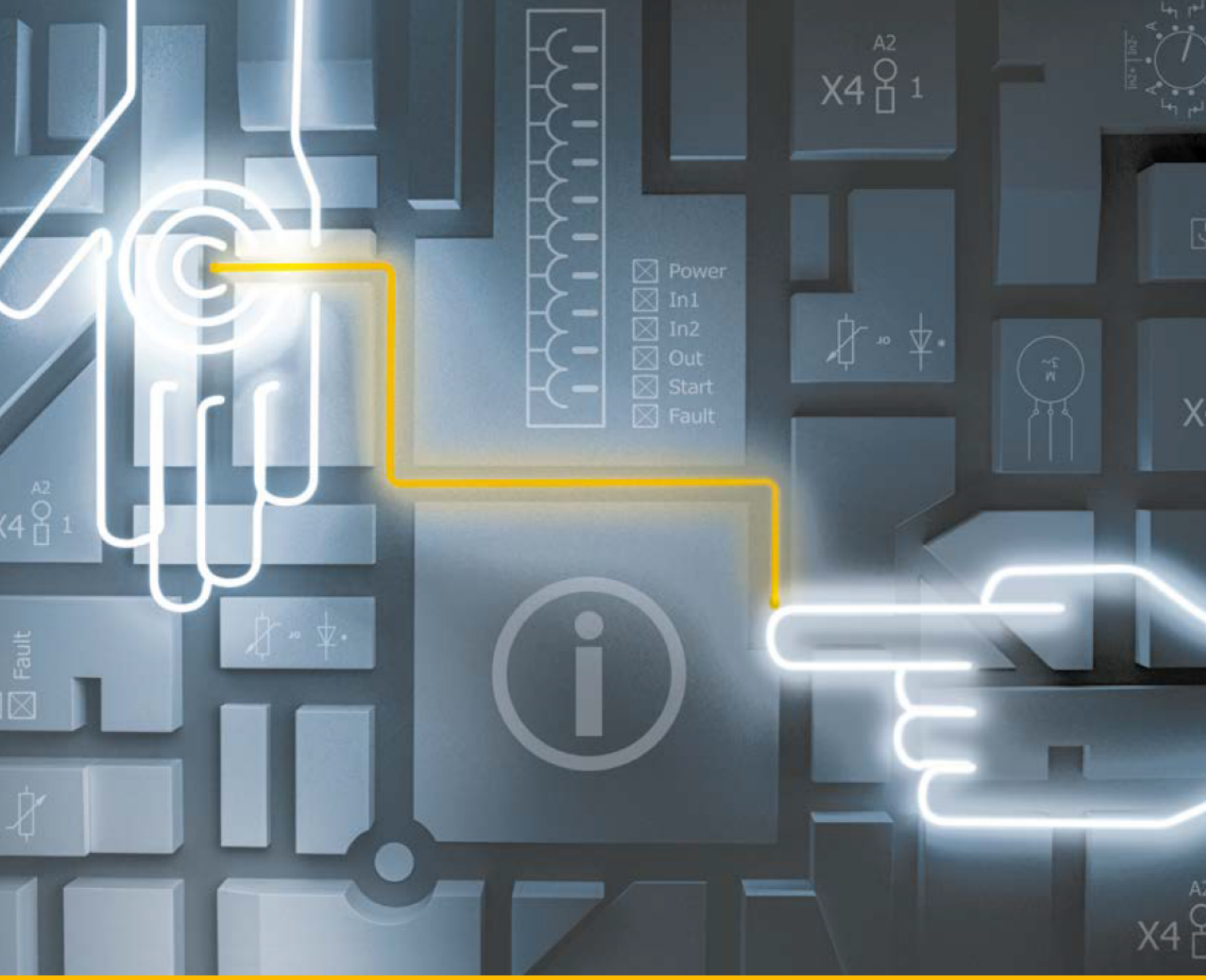

PITreader REST API
PITreader Firmware V2.1.x

PILZ
THE SPIRIT OF SAFETY

Operating Manual-1005365-EN-06
- Control and signal devices

1 Introduction

1.1 Validity of documentation

This documentation is valid for the REST API of the PITreader. It is valid until new documentation is published.

1.2 Using the documentation

This documentation is intended for instruction and should be retained for future reference.

1.3 Definition of symbols

Information that is particularly important is identified as follows:

DANGER!

This warning must be heeded! It warns of a hazardous situation that poses an immediate threat of serious injury and death and indicates preventive measures that can be taken.

WARNING!

This warning must be heeded! It warns of a hazardous situation that could lead to serious injury and death and indicates preventive measures that can be taken.

CAUTION!

This refers to a hazard that can lead to a less serious or minor injury plus material damage, and also provides information on preventive measures that can be taken.

NOTICE

This describes a situation in which the product or devices could be damaged and also provides information on preventive measures that can be taken. It also highlights areas within the text that are of particular importance.

INFORMATION

This gives advice on applications and provides information on special features.

2 Intended use

PITreader REST API is part of the PITreader. The REST API is used for access to the device-internal web server's HTTP end points via a user-created application (external Client).

3 Security

To secure plants, systems, machines and networks against cyberthreats it is necessary to implement (and continuously maintain) an overall industrial security concept that is state of the art.
Perform a risk assessment in accordance with VDI/VDE 2182 or IEC 62443-3-2 and plan the security measures with care. If necessary, seek advice from Pilz Customer Support.

3.1 Implemented security measures

- An external Client can only be connected to the web server via HTTPS.
- If an external Client is connected to the web server via HTTP it will automatically be rerouted to HTTPS.
- Each HTTP request to the web server must be authenticated.

3.2 Required security measures

- Please refer to the security measures required in the operating manual for the PITreader.
- An API token should be handled with the same care as a password. The requirements for passwords can be found in the operating manual for the PITreader.

4 Function description

The PITreader has an API (application programming interface). It is a REST API. With the help of a user-created application (e.g. HMI, web application, user software) an external Client can be connected to the PITreader's web server via the REST API.

To exchange information and data, the external Client must send requests in JSON format to one of the web server's HTTP end points on the PITreader. The HTTP methods GET and POST are supported for this purpose. The web server sends appropriate responses in JSON format to the external Client.

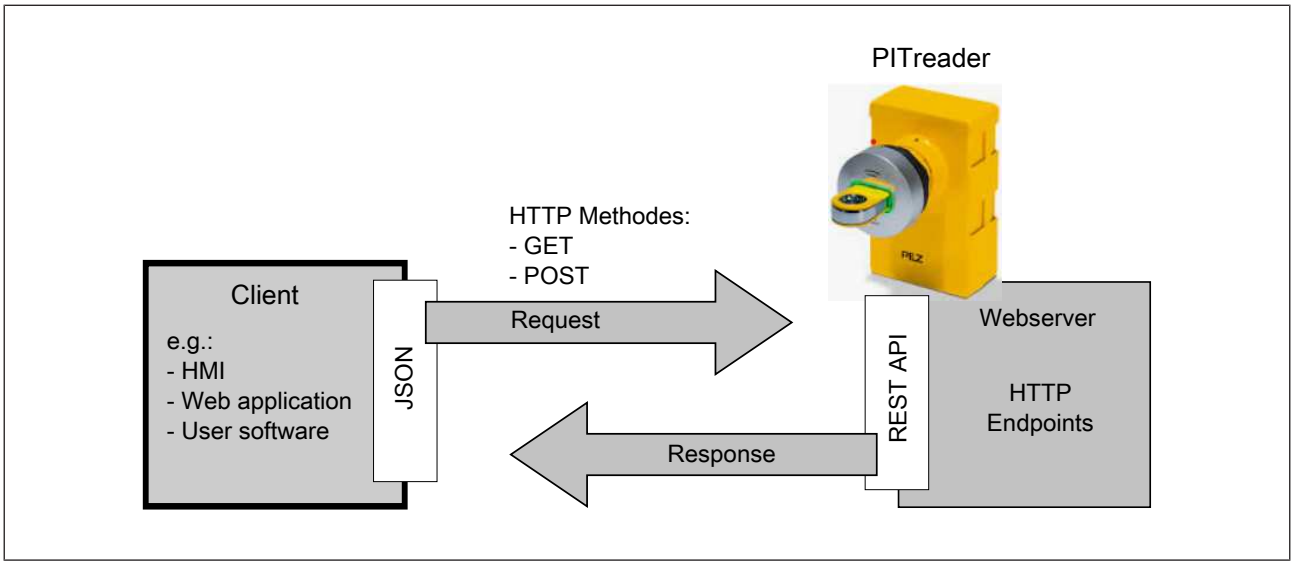

Fig.: Information and data exchange via the REST API (principle)

4.1 Requests from an API Client

An external Client is also known as an API Client. An API Client is in the broadest sense a machine (e.g. PASvisu). Machines on the web server are authenticated via API tokens (API keys); i.e. the API token must be stated in the event of a request from the API Client. An API token is generated in the web application of the PITreader. API Clients can be created and configured in the web application for this purpose.

Creating an API Client

The API Clients can be created in the web application under ***User -> Device users***. An API Client corresponds to a device user with the authentication "API token". As soon as you select the authentication "API token", an API token is generated for the API Client. The API token is a random value 16 Bytes in length; its functionality corresponds to that of a password.

A role must be assigned to the API Client. The role determines the access rights for the API Client.

There are four roles:

- Role name: Administrator
  Role number: 500
- Role name: Device manager
  Role number: 400
- Role name: Transponder manager
  Role number: 200
- Role name: Guest (read access)
  Role number: 10

Access rights for the roles and further information on the device users are described in the PITreader operating manual under "Device users".

Request with authentication via the API token

With each access, the authorisation header with the corresponding API token must be stated in the request. If the authorisation header is missing or invalid, the response will contain the HTTP status code 403 (see HTTP status codes [94]).

The response also contains the HTTP status code 403 when the API token belongs to a device user/API Client whose role number is lower than the stated role number required for the HTTP end point.

Principle structure of a request with authorisation header

<table>
<tr><td colspan="3">Request header</td></tr>
<tr><td><…></td><td colspan="2"><…></td></tr>
<tr><td>GET https://<IP address><Port number><End point>
or
POST https://<IP address><Port number><End point></td><td colspan="2">Request via HTTP method GET or POST (see PITreader's HTTP end points [📖 15])</td></tr>
<tr><td colspan="3">Authorisation header</td></tr>
<tr><td rowspan="3"><Type><Api token></td><td>Type</td><td>Always: Bearer</td></tr>
<tr><td>Api token</td><td>API token (password) generated for the API Client via the web application</td></tr>
<tr><td colspan="2">Example:
Authorisation: Bearer <16 Byte value></td></tr>
<tr><td><…></td><td colspan="2"><…></td></tr>
</table>

4.2 Response format

If the Client requests it, the web server sends a response in JSON format.

The response always consists of the response header, which includes the HTTP status code and the "Content type" response field.

The response header is followed by the body.
In the event of a GET request, the body will consist of the generic response data fields; if the request is OK it will also contain the request-specific response data fields.
In the event of a POST request, the body will consist exclusively of the generic response data fields.

Principle structure of the response

Response Header	
Status code	*<HTTP status code>* Further information is available under HTTP status codes [📖 94].
Content type	Always: application/json
<…>	<…>
Generic response data fields	
msg	"" (empty string) Note: The content may vary, depending on the POST request.
data	null
success	*<Content depends on status code>*
Request-specific response data fields Request-specific response data fields are present exclusively in response to an OK GET request; i.e.: ▸ Response to a request with HTTP method GET ▸ Response header = HTTP status code 200	
<…>	<…>

Structure and content of a response dependent on the HTTP status code

Information on the various response formats can be found in the following sections:

- OK request: See Response format with HTTP status code 200 [📖 11]
- Bad request: See Response format with HTTP status code 400 [📖 11]
- Forbidden request: See Response format with HTTP status code 403 [📖 12]
- Internal error: See Response format with HTTP status code 500 [📖 12]

4.2.1 Response format with HTTP status code 200

In the event of an OK request, the web server responds with HTTP status code 200 (OK) in the response header.

Contents of response header

- HTTP status code: 200
- Content type: application/json
- *<Other>*

Generic data fields in the body

Field name	Data type	Content
msg	STRING	"" (empty string)
data	OBJECT	null
success	BOOLEAN	true

Request-specific data fields in the body

Request-specific data fields are written during requests to the relevant end points.

INFORMATION

The response only contains request-specific data fields in the body when a request with HTTP method GET was sent to the web server and the request was OK (HTTP status code 200).

4.2.2 Response format with HTTP status code 400

In the event of a bad request, the web server responds with HTTP status code 400 (bad request) in the response header.

Contents of response header

- HTTP status code 400
- Content type: application/json
- *<Other>*

Generic data fields in the body

Field name	Data type	Content
msg	STRING	"" (empty string)
data	OBJECT	null
success	BOOLEAN	false

INFORMATION

The content of the generic data field "msg" may vary from the content documented here, depending on the HTTP end point. In this case the content of the generic data field "msg" is documented at the HTTP end point.

4.2.3 Response format with HTTP status code 403

In the event of a forbidden request, the web server responds with HTTP status code 403 (forbidden). HTTP status code 403 means that a request without an existing user session has been sent to the web server or that the API Client does not have the required role number.

Contents of response header

- HTTP status code 403
- Content type: application/json
- *<Other>*

Generic data fields in the body

Field name	Data type	Contents
msg	STRING	"" (empty string)
data	OBJECT	null
success	BOOLEAN	false

4.2.4 Response format with HTTP status code 500

In the event of an internal error, the web server responds with HTTP status code 500 (internal server error).

Contents of response header

- HTTP status code 500
- Content type: application/json
- *<Other>*

Generic data fields in the body

Field name	Data type	Content
msg	STRING	"" (empty string)
data	OBJECT	null
success	BOOLEAN	false

4.3 Compatibility after the PITreader firmware is updated

If the major firmware version (= 1st digit) of the 3-digit firmware version (e.g. 1.5.0) does not change, a PITreader firmware update is downward compatible with regard to the REST API.

With a downward-compatible firmware update, a device with an old firmware version (e.g. 1.5.0) can be replaced by a device with new, downward-compatible firmware (e.g. 1.6.0), without having to make any adjustments with regard to the REST API.

The following applies for downward-compatible PITreader firmware:

- All end points of the "old" REST API are available. However, further end points can be added in the course of development.
- All GET request parameters and all POST request fields on the "old" REST API are available. However, further optional parameters and fields can be added in the course of development.
- If the Client requests it, the web server sends a response in JSON format. The response contains all the data fields of the "old" REST API. However, further fields can be added in the course of development.

4.4 Establishing/terminating a connection between Client and web server

The Client (e.g. HMI, web application, user software) connects to the PITreader's web server by sending an HTTP request to the web server's URL. Following an OK request the connection is again terminated.

Connections to the web server are only possible via HTTPS (standard port 443). If a connection to the web server is established via HTTP (standard port 80), it will be rerouted to HTTPS via HTTP status 301 "Moved Permanently".

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/*<End point>*

- **Parameter**

<IP address>	Web server's IP address Default IP address: 192.168.0.12
<Port number>	Standard port with HTTPS: 443
<End point>	Required end point depending on the HTTP method (GET/POST)

- **Example**

  GET https://192.168.0.12:443/api/status

Response

- The response depends on the request.
- For the example stated see HTTP end point /api/status (GET) [📖 18].

4.5 Synchronisation with external data

There is data on the PITreader (e.g. permission list and block list), which can be updated by an external service (e.g. PIT User Authentication Service). The data is maintained in a database and then synchronised with the PITreader by the external service.

This synchronisation can be monitored, to ensure that it actually takes place. Synchronisation monitoring increases security. See also PITreader operating manual, under "Synchronisation with external data".

- Enable synchronisation monitoring:

  In the POST request at the end point /api/config with the parameter "syncMonitoringEnabled"

- Define synchronisation timeout

  In the POST request at the end point /api/config with the parameter "syncTimeout"

- Restart synchronisation monitoring

  There are two alternatives for restarting monitoring with the defined timeout. In both cases, the API Client requires a role with a role number ≥ 200 (transponder manager).

  – In the GET request at the end point /api/status/monitor with the parameter "syncStatus"
  If the parameter is used, although synchronisation monitoring is not even enabled, then the parameter is ignored.

  – With the HTTP header "X-Sync-Status:1"

  If the HTTP header "X-Sync-Status" is used, although synchronisation monitoring is not even enabled, then the header is ignored.

5 PITreader's HTTP end points

An external Client can send requests to the following web server end points:

HTTP end point	HTTP method	Meaning
General device information		
/api/status	GET	Read general device information (e.g. device name, order number, serial number, firmware version, hardware version, software version)
		See HTTP end point /api/status (GET) [18]
/api/config	GET	Read general device configuration data (e.g. Ethernet parameters)
		See HTTP end point /api/config (GET) [22]
	POST	Define general device configuration data
		See HTTP end point /api/config (POST) [26]
/api/config/coding	GET	Read data for basic coding of device
		See HTTP end point /api/config/coding (GET) [32]
	POST	Define basic coding of a device
		See HTTP end point /api/config/coding (POST) [33]
/api/config/coding/oem	GET	Read data for OEM coding of device
		See HTTP end point /api/config/coding/oem (GET) [35]
	POST	Define OEM coding of a device
		See HTTP end point /api/config/coding/oem (POST) [36]
/api/config/blocklist	GET	Read content of device's block list
		See HTTP end point /api/config/blocklist (GET) [38]
	POST	Create entry in device's block list, change comment of an existing entry or delete an entry
		See HTTP end point /api/config/blocklist (POST) [39]
/api/config/permissionList	GET	Read content of device's permission list
		See HTTP end point /api/config/permissionList (GET) [42]
	POST	Create entry in device's permission list, change permission of an existing entry or delete an entry
		See HTTP end point /api/config/permissionList (POST) [43]

HTTP end point	HTTP method	Meaning
/api/config/ devicegroups	GET	Show names of the device groups to which the device was assigned
		See HTTP end point /api/config/devicegroups (GET) [46]
	POST	Enter name for a device group or change name of a device group
		See HTTP end point /api/config/devicegroups (POST) [47]
Authentication status		
/api/status/ authentication	GET	Read a transponder's current authentication status
		See HTTP end point /api/status/authentication (GET) [49]
Transponder data		
/api/transponder	GET	Read data from the transponder (e.g. group permissions, valid from/until and lock)
		See HTTP end point /api/transponder (GET) [52]
	POST	Write data to the transponder
		See HTTP end point /api/transponder (POST) [54]
/api/transponder/ teachIn	POST	Teach coding identifier to transponder with locked permission area
		See HTTP end point /api/transponder/teachIn (POST) [56]
User data		
/api/config/userData	GET	Read the user data configuration for the transponder (including version information)
		See HTTP end point /api/config/userData (GET) [58]
/api/config/userData/ reset	POST	Delete user data configuration on the PITreader
		See HTTP end point /api/config/userData/reset (POST) [59]
/api/config/userData/ version	POST	Define version of user data configuration
		See HTTP end point /api/config/userData/version (POST) [60]
/api/config/userData/ parameter	POST	Set or change an individual parameter of the user data configuration on the PITreader
		See HTTP end point /api/config/userData/parameter (POST) [61]
/api/transponder/ userData	GET	Read user data that was loaded previously into the RAM of the PITreader
		See HTTP end point /api/transponder/userData (GET) [64]

HTTP end point	HTTP method	Meaning
/api/transponder/ userData/read	POST	Load user data from a transponder into the RAM of the PITreader
		See HTTP end point /api/transponder/userData/read (POST) [67]
/api/transponder/ userData/clear	POST	Reset user data in the RAM of a PITreader
		See HTTP end point /api/transponder/userData/clear (POST) [69]
/api/transponder/ userData/write	POST	Write user data from the RAM of the PITreader to a transponder
		See HTTP end point /api/transponder/userData/write (POST) [70]
/api/transponder/ userData/clearGroup	POST	Delete data for an individual device group in the RAM of a PITreader
		See HTTP end point /api/transponder/userData/clear-Group (POST) [72]
/api/transponder/ userData/ addGroupValues	POST	Add data to a device group in the RAM of the PITreader
		See HTTP end point /api/transponder/userData/ad-dGroupValues (POST) [74]
External authentication		
/api/status/ authentication	POST	External authentication: Set a transponder's authentica-tion status
		See HTTP end point /api/status/authentication (POST) [77]
/api/led	GET	Read LED status and "LED overwrite" settings
		See HTTP end point /api/led (GET) [81]
	POST	Set "LED-overwrite" settings
		See HTTP end point /api/led (POST) [83]
"Single authentication" authentication type		
/api/status/ authentication/ singleAuth	GET	Single authentication: Read status of authentication lock
		See HTTP end point /api/status/authentication/ singleAuth (GET) [85]
Information on the current user		
/api/me	GET	Read information about the current user
		See HTTP end point /api/me (GET) [86]
Status of the PITreader		
/api/status/monitor	GET	Read status of the PITreader and restart synchronisa-tion monitoring
		See HTTP end point /api/status/monitor (GET) [87]

5.1 General device information

5.1.1 HTTP end point /api/status (GET)

Via the HTTP end point /api/status, current status information on the status and properties of the device can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/status

- **Required role number**

  – None, then the response does not contain all possible data fields

  – ≥ 10 (guest), then the response contains all possible data fields

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/status

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of

  – Successful authentication (role number ≥ 10 (guest))

  – HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
hostName	STRING	Hostname/device name	
ipAddress	STRING	Device IP address	
orderNo	STRING	Device's order number	
productType	STRING	Device's product name	
serialNo	NUMBER	Device’s serial number	
macAddress	STRING	Device's MAC address Format: 00:00:00:00:00:00	
seuStatus	BOOLEAN	Status of the connection between PIT m4SEU and PITreader Possible content:	
		true	There is an active connection between a PIT m4SEU and the PITreader.
		false	There is no connection between a PIT m4SEU and the PITreader.

Field name	Data type	Meaning	
coded	BOOLEAN	Status of the basic coding on the PITreader Possible content:	
		true	A coding identifier is stored.
		false	No coding identifier is stored.
codedOem	BOOLEAN	Status of the OEM coding on the PITreader Possible content:	
		true	A coding identifier is stored.
		false	No coding identifier is stored.
transponderAuthenticated	BOOLEAN	Transponder's authentication status Possible content:	
		true	The transponder was authenticated.
		false	The transponder was not authenticated.
fwVersion	STRING	Device's firmware version Format: 00.00.00	
hwVersion	STRING	Hardware version of the PITreader Format: 00.00	
hwVariant	NUMBER	Hardware type of the PITreader	
realTimeClock	STRING	Device’s current date and time (stated in UTC format) Example: 2018-06-28T00:39:53Z	
released	BOOLEAN	True: it is officially released firmware.	
ioPortMode	STRING	Configuration of 24 V I/O port Possible content:	
		input	The 24 V I/O port is configured as an input.
		output	The 24 V I/O port is configured as an output.
ioPortValue	NUMBER	Signal present at the 24 V I/O port Possible content:	
		0	Low signal
		1	High signal
tlsFingerprintVerification	NUMBER	Fingerprint verification code	

- Request-specific data fields in the body in the event of
  - Unsuccessful authentication (without role number)
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
hostName	STRING	Hostname/device name	
ipAddress	STRING	Device IP address	
orderNo	STRING	Device's order number	
productType	STRING	Device's product name	
macAddress	STRING	Device's MAC address Format: 00:00:00:00:00:00	
seuStatus	BOOLEAN	Status of the connection between PIT m4SEU and PITreader Possible content:	
		true	There is an active connection between a PIT m4SEU and the PITreader.
		false	There is no connection between a PIT m4SEU and the PITreader.
coded	BOOLEAN	Status of the basic coding on the PITreader Possible content:	
		true	A coding identifier is stored.
		false	No coding identifier is stored.
codedOem	BOOLEAN	Status of the OEM coding on the PITreader Possible content:	
		true	A coding identifier is stored.
		false	No coding identifier is stored.
transponderAuthenticated	BOOLEAN	Transponder's authentication status Possible content:	
		true	The transponder was authenticated.
		false	The transponder was not authenticated.
hwVariant	NUMBER	Hardware type of the PITreader	
released	BOOLEAN	TRUE: it is officially released firmware.	
ioPortMode	STRING	Configuration of 24 V I/O port Possible content:	
		input	The 24 V I/O port is configured as an input.
		output	The 24 V I/O port is configured as an output.
ioPortValue	NUMBER	Signal present at the 24 V I/O port Possible content:	
		0	Low signal
		1	High signal
tlsFingerprintVerification	NUMBER	Fingerprint verification code	

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.2 HTTP end point /api/config (GET)

Via the HTTP end point /api/config, the device configuration can be read using GET.

Request

- **Schematic representation**

  GET https://<*IP address*>:<*Port number*>/api/config

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
hostName	STRING	Hostname/device name	
location	STRING	Devices's location description	
domain	STRING	Domain name for HTTPS certificates	
ipAddress	STRING	Device IP address	
subnetMask	STRING	Device's subnet mask	
defaultGateway	STRING	Standard Gateway's IP address	
httpPort	NUMBER	HTTP port number Default number: 80	
httpEnabled	BOOLEAN	HTTP status Possible content:	
		true	HTTP port is enabled on the device
		false	HTTP port is not enabled on the device
httpsPort	NUMBER	HTTPS port number Default number: 443	
networkDiscoveryEnabled	BOOLEAN	Network discovery with Multicast DNS (mDNS) Possible content:	
		true	Enabled
		false	Not enabled

Field name	Data type	Meaning	
multicastConfEnabled	BOOLEAN	Network configuration via Multicast protocol Possible content:	
		true	Enabled
		false	Not enabled
sntpEnabled	BOOLEAN	SNTP status Possible content:	
		true	SNTP is enabled on the device
		false	SNTP is not enabled on the device
sntpServer	STRING	SNTP Server's IP address	
sntpPort	NUMBER	SNTP Server's port number	
sntpRefreshRate	NUMBER	Refresh interval Interval at which the PITreader polls the SNTP Server, in minutes	
modbusTcpEnabled	BOOLEAN	Modbus/TCP status Possible content:	
		true	Modbus/TCP Slave is enabled on the device
		false	Modbus/TCP Slave is not enabled on the device
modbusTcpPort	NUMBER	Modbus/TCP Slave's port number	
supportThirdParty Transponders	BOOLEAN	Support third-party transponders	
		true	Yes
		false	No
authenticationMode	NUMBER	Device's authentication mode Possible content:	
		0	External
		1	Transponder data
		2	Permission list
		3	Fixed permission
fixedPermission	NUMBER	Fixed permission for "Fixed permission" authentication mode (Hamming coded, HD9)	
allowExternalOverride	BOOLEAN	Allow external overwrite	
		true	Irrespective of the authentication mode, the permission for the device group and the user data can be overwritten externally via the REST API.
		false	Overwrite is not permitted.

Field name	Data type	Meaning	
syncMonitoringEnabled	BOOLEAN	Synchronisation with external data	
		true	Synchronisation with external data is monitored.
		false	Synchronisation with external data is not monitored.
syncTimeout	NUMBER	Synchronisation timeout in minutes (maximum permitted interval between two synchronisations)	
deviceGroup	NUMBER	Device's device group for authentication mode "1" (transponder data)	
ioPortFunction	NUMBER	Mode of 24 V I/O port Possible content:	
		0	No function
		1	Authentication status (output)
		2	Block authentication (input)
ioPortPermission	NUMBER	If the 24 V I/O port is configured as an output (field "ioPortFunction"=1): Minimum permission of the transponder from which point the output is switched on. If the permission of the transponder is equal to or greater than the set permission, the output assumes the "1" state (Hamming coded, HD9)	
evaluateTimeLimitation	BOOLEAN	Indicates whether start date ("Valid from") and end date ("Valid until") are evaluated by the transponder. Possible content:	
		true	Evaluation activated
		false	Evaluation not activated
timeZone	STRING	Name of time zone Possible content: see Time zones [95]	
logPersonalData	BOOLEAN	Recording of personal data in diagnostic list and diagnostic log Possible content:	
		true	Personal data is recorded
		false	Personal data is not recorded
authenticationType	NUMBER	Active authentication type Possible content:	
		0	"Basic" authentication type
		1	"Single authentication" authentication type
		2	"2-person rule" authentication type

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.3 HTTP end point /api/config (POST)

Via the HTTP end point /api/config, various parameters for the device configuration can be defined using POST.

Request

- Schematic representation

  POST https://<*IP address*>:<*Port number*>/api/config

- **Required role number**

  ≥ 400 (device manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

  Parameters that may be used by API Clients with role number 400 or 500:

  location <*Location description*> — Device's location description
  (Data type: STRING)
  - Number of characters [Byte]: 0 … 47
  - Valid UTF-8 characters or printable ASCII characters
  - Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 47.

  deviceGroup <*Number*> — Enter device's device group for authentication mode "1" (transponder data)
  (Data type: NUMBER)
  - Valid numbers:
    0 … 9999

  evaluateTimeLimitation <*true/false*> — Evaluate the transponder's start date ("Valid from") and end date ("Valid until")
  (Data type: BOOLEAN)

false	Do not activate evaluation
true	Activate evaluation

  timeZone <*Name*> — Name of time zone
  (Data type: STRING)
  - Number of characters [Byte]: 0 … 31
  - It must be a valid time zone (see Time zones [📖 95]).

  realTimeClock <*Date*> — Date and time
  (Data type: STRING)
  - String in accordance with RFC3339 section-5.6, date and time in UTC
  - min. 01.01.2000 00:00 UTC
  - Format: 2018-06-28T00:00:00Z

  Note:

  All the parameters transferred must be valid and at least one parameter must be transferred.

Parameters that may only be used by API Clients with role number 500:

hostName <*Name*>
Hostname/name of device
(Data type: STRING)
- Number of characters [Byte]: 1 … 63
- Valid characters: a-z, A-Z, 0-9, "-"
- The first character must be a letter.
- The last character may not be a hyphen.

domain <*Name*>
Domain name for HTTPS certificates
(Data type: STRING).
- Number of characters [Byte]: 0 … 63
- Valid characters: a-z, A-Z, 0-9, "-", "."
- The first character must be a letter.
- The last character may not be a hyphen nor a dot.

ipAddress <*IP address*>
IP address of device/host
(Data type: STRING)
- Valid host IP address, not 0.0.0.0/8, 127.0.0.0/8 and 224.0.0.0/4

subnetMask <*Mask*>
Device’s subnet mask
(Data type: STRING)
- Subnet mask in binary format
- The subnet mask must start with a 1 and end with a 0.
- It may only switch once from 1 to 0.

defaultGateway <*Address*>
IP address of the Standard Gateway
(Data type: STRING)
- Valid host IP address, not 0.0.0.0/8, 127.0.0.0/8 and 224.0.0.0/4
- The IP address must be in the same subnet as the host IP address ("ipAddress" parameter).

httpPort <*Number*>
HTTP port number
(Data type: NUMBER)
- Default port: 80
- Valid numbers: 1 … 65535
- Port may not be identical to "httpsPort" or "modbusPort".

httpEnabled <*true/false*>
HTTP
(Data type: BOOLEAN)

true	Enable HTTP
false	Disable HTTP

httpsPort <*Number*>	HTTPS port number (Data type: NUMBER) ▸ Default port: 443 ▸ Valid numbers: 1 … 65535 ▸ Port may not be identical to "httpPort" or "modbusPort".
networkDiscoveryEnabled <*true/false*>	Network discovery with Multicast DNS (mDNS) (Data type: BOOLEAN) true Enable false Disable
multicastConfEnabled <*true/false*>	Network configuration via Multicast protocol (Data type: BOOLEAN) true Enable false Disable
sntpEnabled <*true/false*>	SNTP (Data type: BOOLEAN) true Enable SNTP false Disable SNTP
sntpServer <*Address*>	SNTP Server's IP address (Data type: STRING) ▸ Valid host IP address, not 0.0.0.0/8, 127.0.0.0/8 and 224.0.0.0/4
sntpPort <*Port*>	SNTP Server's port number (Data type: NUMBER) ▸ Valid numbers: 1 … 65535
sntpRefreshRate <*Minutes*>	Refresh interval Interval at which the PITreader polls the SNTP Server, in minutes (Data type: NUMBER) ▸ Valid values: 5 … 10080
modbusTcpEnabled <*true/false*>	Modbus/TCP (Data type: BOOLEAN) true Enable Modbus/TCP Slave on the device false Disable Modbus/TCP Slave on the device
modbusTcpPort <*Port*>	Modbus/TCP Slave's port number (Data type: NUMBER) ▸ Valid numbers: 1 … 65535 ▸ Port may not be identical to "httpPort" or "httpsPort".

supportThirdPartyTransponders *<true/false>*
Support third-party transponders
(Data type: BOOLEAN)

true	Support third-party transponders
false	Do not support third-party transponders

authenticationMode *<Mode>*
Device's authentication mode
(Data type: NUMBER)

0	External
1	Transponder data
2	Permission list
3	Fixed permission

fixedPermission *<Permission>*
Fixed permission for "Fixed permission" authentication mode
(Data type: NUMBER)

- Valid permissions:
  0 ... 64
- Hamming coded, HD9

allowExternalOverride *<true/false>*
Allow external overwrite
(Data type: BOOLEAN)

true	Irrespective of the authentication mode, the permission for the device group and the user data can be overwritten externally via the REST API.
false	Overwrite is not permitted.

syncMonitoringEnabled *<true/false>*
Synchronisation with external data
(Data type: BOOLEAN)

true	Monitor synchronisation with external data
false	Do not monitor synchronisation with external data

syncTimeout *<Minutes>*
Synchronisation timeout in minutes (maximum permitted interval between two synchronisations)
(Data type: NUMBER)

- Valid values:
  5 … 4320

logPersonalData *<true/false>*
Recording of personal data in diagnostic list and diagnostic log
(Data type: BOOLEAN)

true	Record personal data
false	Do not record personal data

ioPortFunction *<Value>*
Mode of 24 V I/O port
(Data type: NUMBER)

0	No function
1	Authentication status (output)
2	Block authentication (input)

ioPortPermission <*Permission*>	If the 24 V I/O port is configured as an output (parameter "ioPortFunction"=1): Minimum permission of the transponder from which point the output is to be switched on. If the permission of the transponder is equal to or greater than the set permission, the output assumes the "1" state. (Data type: NUMBER) ▸ Valid permissions: 1 ... 64 ▸ Hamming coded, HD9
authenticationType <*Value*>	Authentication type (Data type: NUMBER) 0 "Basic" authentication type 1 "Single authentication" authentication type 2 "2-person rule" authentication type

Note:

All the parameters transferred must be valid and at least one parameter must be transferred.

▸ **Example**

```
POST https://192.168.0.12/api/config
{
    "location": "Ostfildern A20",
    "deviceGroup": 10,
    "evaluateTimeLimitation": false
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

In contrast to the content documented under Response format with HTTP status code 200 [📖 11], the generic data field "data" can contain more detailed information.

Possible additional content of the data field "data":

- "data": { "reboot": true }: A reboot was performed due to the change.
- "data": { "reboot": false }: A reboot was not performed due to the change.

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible content of the generic data field "msg"

- Cause of error: incorrect parameters

  Content in the generic data field "msg":

  – API.errorLocation

  – API.errorDeviceGroup

  – API.errorEvaluateTimeLimitation

  – API.errorTimeZone

  – API.errorRealTimeClock

5.1.4 HTTP end point /api/config/coding (GET)

Via the HTTP end point /api/config/coding, data for basic coding of the device can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/config/coding

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config/coding

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
activated	BOOLEAN	Status of the device's basic coding (basic identifier)	
		true	Basic coding set
		false	Basic coding not set
checksum	STRING	Check sum for basic coding ▸ 16 Byte HEX string ▸ Format: 00:00.00:00	
comment	STRING	Comment about basic identifier	

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.5 HTTP end point /api/config/coding (POST)

Via the HTTP end point /api/config/coding, the basic coding of a device can be defined using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/config/coding

- **Required role number**

  ≥ 400 (device manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

comment <*Comment*>	Comment about the basic identifier (STRING data type) ▸ Number of characters [Byte]: 0 … 63 ▸ Valid UTF-8 characters ▸ Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 63.
identifier <*Basic ID*>	Basic identifier (STRING data type) ▸ Number of characters [Byte]: 0 … 63 ▸ Valid UTF-8 characters ▸ Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 63.

  Note: At least one of the parameters must be transferred. It is not strictly necessary to transfer the basic identifier if only a comment about the basic identifier is to be stored or the comment is to be changed.

- **Example**

```
POST https://192.168.0.12/api/config/coding
{
    "identifier": "Secret Coding ID 123",
    "comment": "My coding #1"
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

5.1.6 HTTP end point /api/config/coding/oem (GET)

Via the HTTP end point /api/config/coding/oem, data for OEM coding of the device can be read using GET.

Request

- **Schematic representation**

  GET https://<IP address>:<Port number>/api/config/coding/oem

- **Required role number**

  ≥ 500 (Administrator)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config/coding/oem

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
activated	BOOLEAN	Status of the OEM coding on the device (OEM identifier)	
		true	OEM coding set
		false	OEM coding not set
checksum	STRING	Check sum for OEM coding ▸ 16 Byte HEX string ▸ Format: 00:00.00:00	
comment	STRING	Comment about OEM identifier	

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.7 HTTP end point /api/config/coding/oem (POST)

Via the HTTP end point /api/config/coding/oem, the OEM coding of a device can be defined using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/config/coding/oem

- **Required role number**

  ≥ 500 (Administrator)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

  comment <*Comment*>
  Comment about OEM identifier
  (STRING data type)
  - Number of characters [Byte]: 0 … 63
  - Valid UTF-8 characters
  - Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 63.

  oldIdentifier <*OEM identifier*>
  OEM identifier currently on the device. If no OEM identifier is present, then an empty string must be transferred.
  (Data type: STRING)
  - Number of characters [Byte]: 0 … 63
  - Valid UTF-8 characters
  - Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 63.

  newIdentifier <*OEM identifier*>
  New OEM identifier, to which the device is to be set. If an existing OEM identifier is to be deleted, then an empty STRING must be transferred.
  (Data type: STRING)
  - Number of characters [Byte]: 0 … 63
  - Valid UTF-8 characters
  - Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 63.

  Note: The parameters *oldIdentifier* and *newIdentifier* must always both be transferred. The additional transfer of the *comment* parameter is optional.
  If only a comment about the OEM identifier is to be stored or the comment is to be changed, then the parameters *oldIdentifier* and *newIdentifier* are omitted.

Example

```
POST https://192.168.0.12/api/config/coding/oem
{
    "oldIdentifier": "Secret Coding ID 123",
    "newIdentifier": "Secret Coding ID 456",
    "comment": "My coding #1"
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

5.1.8 HTTP end point /api/config/blocklist (GET)

Via the HTTP end point /api/config/blocklist, the content of the device's block list can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/config/blocklist

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config/blocklist

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name		Data type	Meaning
items		ARRAY OF OBJECT	Block list with locked transponders
	id	STRING	Security ID of the locked transponder
	comment	STRING	Comment about locked transponder

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.9 HTTP end point /api/config/blocklist (POST)

Via the HTTP end point /api/config/blocklist, an entry can be created in the device's block list (lock transponder), the comment about an existing entry can be changed or an entry can be deleted using POST.

Request

- Schematic representation

  POST https://<IP address>:<Port number>/api/config/blocklist

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

  Create entry:

action <*Type of request*>	Type of request: create (STRING data type)
id <*Security ID*>	Transponder's security ID (STRING data type)
data	Transfer a comment (OBJECT data type)
comment <*Comment*>	Comment about locked transponder (Optional) (STRING data type) ▸ Number of characters [Byte]: 0 … 32 ▸ Valid UTF-8 characters ▸ Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 32.

  Note: The parameters "action", id" and "data" must always be transferred.

  Note:

  - The block list may contain a maximum of 1000 entries.
  - A security ID may only occur once within the block list.

  Change the comment for an entry:

action <*Type of request*>	Type of request: edit (STRING data type)
id <*Security ID*>	Transponder's security ID (STRING data type)
data	Transfer a comment (OBJECT data type)
comment <*Comment*>	Changed comment about locked transponder ▸ Number of characters [Byte]: 0 … 32 ▸ Valid UTF-8 characters ▸ Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 32.

  Note: The parameters "action", id", "data" and "comment" must always be transferred.

Delete entry:

action <*Type of request*>	Type of request: delete (STRING data type)
id <*Security ID*>	Transponder's security ID (STRING data type)

Note: The parameters "action" and "id" must always be transferred.

Example

```
POST https://192.168.0.12/api/config/blocklist
{
        "action": "create",
        "id": "309A68BACCA44C30",
        "data":
        {
            "comment": "Lost 2021-01-19"
        }
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible contents of the generic data field "msg":

- Cause of error: operation not permitted or invalid parameters

  Content in the generic data field "msg":

  – API.errorMaximumSize: The block list already contains the maximum permitted number of entries.

  – API.errorDuplicateSecurityId: The block list already contains the security ID.

  – API.errorInvalidSecurityId: The security ID is invalid.

5.1.10 HTTP end point /api/config/permissionList (GET)

Via the HTTP end point /api/config/permissionList, the content of the device's permission list can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/config/permissionList

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config/permissionList

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name		Data type	Meaning
items		ARRAY OF OBJECT	Entries in the permission list
	id	STRING	Transponder's security ID
	permission	NUMBER	Permission of the transponder (Hamming coded, HD9)

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.11 HTTP end point /api/config/permissionList (POST)

Via the HTTP end point /api/config/permissionList, an entry can be created in the device's permission list, the permission of an existing entry can be changed or an entry can be deleted using POST.

Request

- Schematic representation

  POST https://<IP address>:<Port number>/api/config/permissionList

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

  Create entry:

action <*Type of request*>	Type of request: create (STRING data type)
id <*Security ID*>	Transponder's security ID (STRING data type)
data	Transfer of permission (OBJECT data type)
permission <*Permission*>	Permission of the transponder, Hamming coded (NUMBER data type)

  Note: The parameters "action", id", "data" and "permission" must always be transferred.

  Note:

  - The permission list may contain a maximum of 1000 entries.
  - A security ID may only occur once in the permission list.

  Change permission of an entry:

action <*Type of request*>	Type of request: edit (STRING data type)
id <*Security ID*>	Transponder's security ID (STRING data type)
data	Transfer of permission (OBJECT data type)
permission <*Permission*>	Changed permission of the transponder, Hamming coded (NUMBER data type)

  Note: The parameters "action", id", "data" and "permission" must always be transferred.

  Delete entry:

action <*Type of request*>	Type of request: delete (STRING data type)
id <*Security ID*>	Transponder's security ID (STRING data type)

  Note: The parameters "action" and "id" must always be transferred.

▸ **Example**

```
POST https://192.168.0.12/api/config/permissionList
{
        "action": "create",
        "id": "309A68BACCA44C30",
        "data":
        {
            "permission": 116684
        }
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible contents of the generic data field "msg":

▸ Cause of error: operation not permitted or invalid parameters

Content in the generic data field "msg":

– API.errorMaximumSize: The permission list already contains the maximum permitted number of entries.

– API.errorDuplicateSecurityId: The permission list already contains the security ID.

– API.errorInvalidSecurityId: The security ID is invalid or is not included in the permission list (the latter only with the request "edit" or "delete").
– API.errorPermission: The permission is invalid.

5.1.12 HTTP end point /api/config/devicegroups (GET)

Via the HTTP end point /api/config/devicegroups, the names of the device groups to which the device has been assigned can be displayed using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/config/devicegroups

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config/devicegroups

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning
items	ARRAY of OBJECT	List containing the names of the device groups to which the device has been assigned
name *<Name_G0>*	STRING	Name of device group G0 or number of device group
…	…	…
name *<Name_G31>*	STRING	Name of device group G31 or number of device group

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.1.13 HTTP end point /api/config/devicegroups (POST)

Via the HTTP end point /api/config/devicegroups, a name for a device group can be entered or the name of a device group can be changed using POST.

Note: The device must be assigned to the device group; i.e. it is not possible to assign the device to a device group via this HTTP end point.

Request

- Schematic representation

  POST https://<IP address>:<Port number>/api/config/devicegroups

- **Required role number**

  ≥ 400 (device manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

number <*Index*>	Number of the device group for which a name is to be created or whose name is to be changed (NUMBER data type) Index: 0 … 31
name <*Name*>	Name of device group (STRING data type) ▸ Number of characters [Byte]: 0 … 47 ▸ Valid UTF-8 characters ▸ Entering UTF-8 characters that need more than 1 Byte reduces the maximum permitted number of characters of 47.

- **Example**

```
POST https://192.168.0.12/api/config/devicegroups
{
    "number": 0,
    "name": "Lathes, Hall 15"
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible contents of the generic data field "msg":

- Cause of error: invalid parameter

  Content in the generic data field "msg":

  – "" (empty string): no data received

  – API.errorDevicegroupFormat: incorrect content in "name"

  – API.errorInvalidDevicegroupIndex: invalid value in "number"

5.2 Authentication status

5.2.1 HTTP end point /api/status/authentication (GET)

Via the HTTP end point /api/status/authentication, current information on a transponder's authentication status can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/status/authentication

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/status/authentication

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format.

Field name	Data type	Meaning	
authenticated	BOOLEAN	Transponder's authentication status Possible content:	
		true	The transponder was authenticated.
		false	The transponder was not authenticated.
permission	NUMBER	Authenticated permission	
authenticationStatus	NUMBER	Status of authentication process Possible content:	
		0	No transponder
		1	Process completed
		2	Waiting for external authentication

Field name	Data type	Meaning	
failureReason	NUMBER	Reason for failed authentication Possible content:	
		0	No error
		1	No transponder positioned
		2	Permission "0" The transponder has no permissions for device groups.
		3	The transponder is invalid. The current date is outside the transponder's validity period (start date/end date).
		4	The transponder is included in the block list.
		5	No permission has been stored for the security ID yet. ("External" authentication mode)
		6	Authentication is locked by the 24 V I/O port.
		7	The "Single authentication" authentication type is configured and authentication is locked by another registered transponder.
		8	The "2-person rule" authentication type is configured and a second transponder is required for authentication.
		9	There is no permission in the permission list for the security ID. ("Permission list" authentication mode)
		10	Authentication is locked because a synchronisation timeout occurred.
securityId	STRING	Security ID	
orderNo	NUMBER	Order number of the transponder	
salesVersion	STRING	Suffix of the order number (distribution version)	
serialNo	STRING	Transponder's serial number	
transponderUid	STRING	Transponder UID	
userData	ARRAY of OBJECT	User data	

Field name		Data type	Meaning
	id	NUMBER	Parameter ID
	numericValue	NUMBER	Numeric value of the parameter (optional) Note: The data field is only available when a numeric value is concerned (type ID 10 … 15 or 30).
	stringValue	STRING	String values of the parameter (optional) Note: The data field is only available when a parameter with type ID 1 (STRING) or type ID 20 (DATETIME) is concerned.

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.3 Transponder data

5.3.1 HTTP end point /api/transponder (GET)

Via the HTTP end point /api/transponder, data from a transponder such as group permission, validity (valid from/until) and lock, for example, can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/transponder

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/transponder

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of

  – HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning
teachInId	STRING	Unique transponder identifier, which is needed for coding. In the event of a POST request, this identifier must be transferred to the HTTP end point /api/transponder/teachIn in the parameter "teachInID" (see parameter "teachInID <*String*>" under HTTP end point /api/transponder/teachIn (POST) [📖 56]).
securityId	STRING	Transponder's security ID
transponderUid	STRING	Transponder UID
permissions	ARRAY of NUMBER	Permission for the device group (device groups 0 … 31, Hamming-coded)

Field name	Data type	Meaning	
timeLimitationStart	STRING	Start date for the validity of the transponder (stated in UTC format) Example: 2019-06-01T08:30:59Z Possible content:	
		<Date>	"Valid from" date
		"" (empty string)	No start date (default value)
timeLimitationEnd	STRING	End date for the validity of the transponder (stated in UTC format) Example: 2019-06-30T00:00:00Z Possible content:	
		<Date>	"Valid until" date
		"" (empty string)	No end date (default value)
lockedPermissions	BOOLEAN	Permission status Possible content:	
		true	Permissions locked
		false	Permissions not locked
codingLock	BOOLEAN	Limit transponder to identically coded PITreaders Possible content:	
		0	Limit not activated (default value)
		1	Limit activated

If there is no transponder in the read area, then the contents of the fields are set to their default values.

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [10].

5.3.2 HTTP end point /api/transponder (POST)

Via the HTTP end point /api/transponder, data such as group permission and validity (valid from/until), for example, can be set on a transponder using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/transponder

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

permissions <*permission*>	Hamming-coded permissions for groups 0 … 31 (Data type: ARRAY of NUMBER) ▸ An array with exactly 32 elements must be transferred. ▸ Each element must contain a valid Hamming code for the permission.
timeLimitationStart <*Date/empty*	"Valid from" date for the transponder (Data type: STRING) Date: Start date String in accordance with RFC3339 section-5.6, date and time in UTC (min. 01.01.2000 00:00 UTC, format 2018-06-28T00:00:00Z). Before writing to a transponder, the time component is always set to "00:00:00". Empty: Deletes the start date on the transponder
timeLimitationEnd <*Date/empty*	"Valid until" date for the transponder (Data type: STRING) Date: End date String in accordance with RFC3339 section-5.6, date and time in UTC (min. 01.01.2000 00:00 UTC, format 2018-06-28T00:00:00Z). Before writing to a transponder, the time component is always set to "00:00:00". Empty: Deletes the end date on the transponder
codingLock <*true/false*>	Limit transponder to identically coded PITreaders (Data type: BOOLEAN) false: Limit is not activated. (default value) true: Limit is activated.

Note: Start date, end date and limit can be set using the parameters "timeLimitationStart", "timeLimitationEnd" and "codingLock", irrespective of "lockedPermissions" (see HTTP end point /api/transponder (GET) [📖 52]).

Example

```
POST https://192.168.0.12/api/transponder
{
    "permissions": [ 511, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 ],
    "timeLimitationStart": "",
    "timeLimitationEnd": "2025-12-31T00:00:00Z",
    "codingLock": false
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

5.3.3 HTTP end point /api/transponder/teachIn (POST)

Via the HTTP end point /api/transponder/teachIn, a transponder can be taught a coding identifier using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/transponder/teachIn/

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

teachInId <*String*>	Transponder's unique identifier, which is reported back to the end point /api/transponder in the "teachInID" field in response to a GET request. (Data type: STRING). Note: The POST request is only executed if the string in the "teachInId" parameter and the transponder's identifier match (see "teachInID" field under HTTP end point /api/transponder (GET) [52]).

- **Example**

```
POST https://192.168.0.12/api/transponder/teachIn
{
    "teachInId": "5B7111F84328AB81"
}
```

Response

General information can be found under Response format [10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [10].

5.4 User data

Detailed information on the user data is available in the operating manual of the PITreader.

The diagram below shows an overview of the mode of operation of requests to HTTP end points for user data.

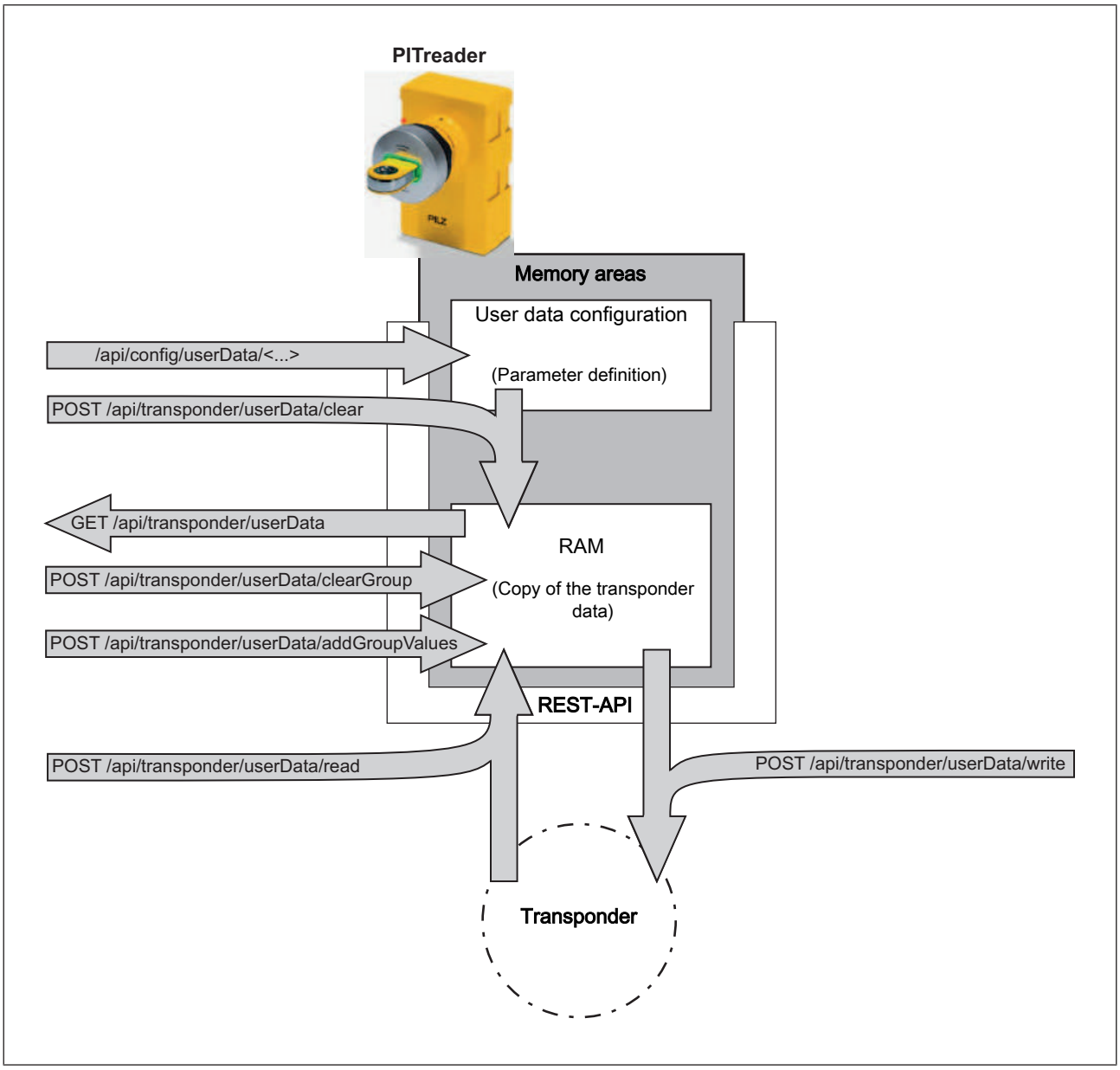

Fig.: Mode of operation of requests to the HTTP end points for user data

5.4.1 HTTP end point /api/config/userData (GET)

Via the HTTP end point /api/config/userData, the user data configuration for the transponder, including version information, can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/config/userData

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/config/userData

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of

  – HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning
version	NUMBER	Version number of the user data configuration Possible content: 1 … 65535
comment	STRING	Comment about the version of the user data configuration Number of characters [Byte]: 0 … 63
parameters	ARRAY of OBJECT	Parameters
id	NUMBER	Parameter ID Possible content: 1 … 65535
name	STRING	Parameter name UTF-8 string with 1 … 63 characters
type	NUMBER	Type ID (= ID of data type)
size	NUMBER	Maximum number of characters for a parameter of the "STRING" data type (optional) Note: The data field is only available on a parameter with type ID 1 (STRING).

5.4.2 HTTP end point /api/config/userData/reset (POST)

Via the HTTP end point /api/config/userData/clear, the user data configuration on the PITreader can be deleted using POST. Once deleted, an empty user data configuration is created; i.e.:

- The data field with the version number of the user data configuration is "0".
- The data field with the comment about the version number is empty.
- The user data configuration is deleted.

Request

- **Schematic representation**

  POST https://*<IP address>:<Port number>*/api/config/userData/reset

- **Required role number**

  ≥ 400 (device manager)

- **Parameters**

  None

- **Example**

  POST https://192.168.0.12/api/config/userData/reset
  <empty / no body>

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

5.4.3 HTTP end point /api/config/userData/version (POST)

The version of the user data configuration for the transponder can be defined via the HTTP end point /api/config/userData/version.

Request

Schematic representation

POST https://<IP address>:<Port number>/api/config/userData/version

Required role number

≥ 400 (device manager)

Parameter

The parameters must be transferred in the body of the request in JSON format.

version <*Number*>	Version number of the user data configuration. (Data type: NUMBER)
	Number: 1 … 65535
comment <*Comment*>	Comment about the version (Data type: STRING)
	Comment: UTF-8 string with 0 … 63 characters

Example

```
POST https://192.168.0.12/api/config/userData/version
{
    "version": 1,
    "comment": "First draft"
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

5.4.4 HTTP end point /api/config/userData/parameter (POST)

An individual parameter of the user data configuration on the PITreader can be reset or changed via the HTTP end point /api/config/userData/parameter.

Note: Via the end point /transponder/userData/clear, changed parameters on the transponder must be applied using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/config/userData/parameter

- **Required role number**

  ≥ 400 (device manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

id <*Number*>	Parameter ID (Data type: NUMBER) Number: 1 … 65535
name <*Name*>	Parameter name (Data type: STRING) Name: UTF-8 string with 1 … 63 characters
type <*Type ID*>	Type ID (= ID of data type) (Data type: NUMBER) Type ID: valid entry see [*1]
size <*Data length*>	Maximum number of characters for a parameter of the "STRING" data type (optional) (Data type: NUMBER) Data length: 2 … 255 (see [*1]) Note: "size" only needs to be transferred in the case of a parameter with type ID 1 (STRING). The number of characters (data length in Bytes) selected must be 1 Byte greater than the number of characters required; i.e.: size = <*Number of characters in Bytes*> + 1 Byte (= terminating character '\0') $size_{max}$ <= 255

Note:

- If the transferred parameter ID is not yet available in the user data configuration, then the parameter is created from new in the user data configuration.
- If the transferred parameter ID is already available in the user data configuration, then the parameter is updated accordingly.

- If a pre-defined parameter ID is used, the data type must meet the relevant specification.
  Examples:
  - ID 1 must always be "type" = 30
  - ID 2 and ID 3 must always be "type" = 20.

  Please refer to the information on system parameters in the operating manual for the PITreader.

- **Example**

```
POST https://192.168.0.12/api/config/userData/parameter
{
    "id": 1,
    "name": "Permission",
    "type": 30
}
```

[*1]

Type-ID	Name	Data length	Value range	Initial value
1	STRING	2 … 255 Byte		Empty STRING
10	INT8U	1 Byte	0 ... 255	0
11	INT8S	1 Byte	-128 ... 127	0
12	INT16U	2 Byte	0 ... 65535	0
13	INT16S	2 Byte	-32768 ... 32767	0
14	INT32U	4 Byte	0 ... 4294967295	0
15	INT32S	4 Byte	-2147483648 ... 2147483647	0
20	DATETIME	4 Byte		Empty time/date
30	PERMISSION	4 Byte	0 ... 64 (Hamming-coded)	0

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible contents of the generic data field "msg":

- Cause of error: a reserved parameter ID with the wrong type ID was used or incorrect information used in the "size" data field.

  Content in the generic data field "msg":

  – API.errorPredefinedId

- Non-specific cause of error

  Content in the generic data field "msg":

  – "" (empty string)

5.4.5 HTTP end point /api/transponder/userData (GET)

Via the HTTP end point /api/transponder/userData, the user data that was previously loaded into the RAM of the PITreader can be read using GET.

Note: Before reading the user data, the data must either be read from a transponder (see HTTP end point /api/transponder/userData/read (POST) [📖 67]) or an empty initialisation must be performed based on the parameters of the PITreader (see HTTP end point /api/transponder/userData/clear (POST) [📖 69]).

Request

- **Schematic representation**

  GET https://<IP address>:<Port number>/api/transponder/userData

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/transponder/userData

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of

  – HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning
parameterDefinition	OBJECT	Parameter definition
version	NUMBER	Version of parameter definition Possible content: 0 … 65535
parameters	ARRAY of OBJECT	Parameters that are stored on the transponder
id	NUMBER	Parameter ID Possible content: 1 … 65535
type	NUMBER	Type ID (= ID of data type) Possible type ID see [*1]
size	NUMBER	Maximum number of characters for a parameter of the STRING data type (optional) Note: The data field is only available on a parameter with type ID 1 (STRING). Possible data length see [*1]
groups	ARRAY of OBJECT	Parameter data that is grouped by device group and is stored on the transponder

Field name	Data type	Meaning
deviceGroup	NUMBER	Device group for which the value of the parameter is stored (optional) Note: If the value of the parameter is valid for all device groups (default value), then the "deviceGroup" data field is not available. Possible content: 0 … 9999
values	ARRAY of OBJECT	Parameter data
id	NUMBER	Parameter ID
numericValue	NUMBER	Numeric value of the parameter (optional) Note: The data field is only available when a numeric value is concerned (type ID 10 … 15 or 30). Possible value range see [*1]
stringValue	STRING	String values of the parameter (optional) Note: The data field is only available when a parameter with type ID 1 (STRING) or type ID 20 (DATETIME) is concerned. Possible content:

Type ID	Contents
1	UTF-8 string, see data length [*1]
20	String in accordance with RFC3339 section-5.6, date and time in UTC ▸ min. 01.01.2000 00:00 UTC ▸ Format: 2018-06-28T00:00:00Z ▸ An empty string corresponds to an unset date.

[*1]

Type-ID	Name	Data length	Value range	Initial value
1	STRING	2 … 255 Byte		Empty STRING
10	INT8U	1 Byte	0 ... 255	0
11	INT8S	1 Byte	-128 ... 127	0
12	INT16U	2 Byte	0 ... 65535	0
13	INT16S	2 Byte	-32768 ... 32767	0
14	INT32U	4 Byte	0 ... 4294967295	0
15	INT32S	4 Byte	-2147483648 ... 2147483647	0
20	DATETIME	4 Byte		Empty time/date
30	PERMISSION	4 Byte	0 ... 64 (Hamming-coded)	0

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.4.6 HTTP end point /api/transponder/userData/read (POST)

Via the HTTP end point /api/transponder/userData/read, the user data on a transponder can be loaded into the RAM of the PITreader using POST. If the RAM of the PITreader already contains user data, then this will be overwritten.

Request

- **Schematic representation**

  POST https://*<IP address>:<Port number>*/api/config/userData/read

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  POST https://192.168.0.12/api/transponder/userData/read
  <empty / no body>

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

- Content of the generic data field "msg": "" (empty string)
- Cause of error: transponder is not positioned

Response in the event of a bad request with HTTP status code 500

- Content of the generic data field "msg": "" (empty string)
- Cause of error: Could not read data from the transponder
  - No data available on the transponder or
  - The data on the transponder is invalid

Remedy:
Use the clear end point (see HTTP end point /api/transponder/userData/clear (POST) [📖 69]).

5.4.7 HTTP end point /api/transponder/userData/clear (POST)

Via the HTTP end point /api/transponder/userData/clear, the user data in the RAM of a PITreader can be reset using POST.

Note: On the basis of the parameters for the user data configuration on the PITreader, the memory area for user data on the transponder is reinitialised; i.e. a copy is made of the parameters and the parameter data for all parameters is deleted.

Request

- **Schematic representation**

  POST https://*<IP address>:<Port number>*/api/transponder/userData/clear

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameters**

  None

- **Example**

  POST https://192.168.0.12/api/transponder/userData/clear
  <empty / no body>

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

- Content of the generic data field "msg": "" (empty string)
- Cause of error: transponder is not positioned

5.4.8 HTTP end point /api/transponder/userData/write (POST)

Via the HTTP end point /api/transponder/userData/write, the user data from the RAM of the PITreader can be written to a transponder using POST.

Note: Before you can execute the request, you will need to either reset the user data in the RAM of the PITreader (see HTTP end point /api/transponder/userData/clear (POST) [69]) or load the user data from the transponder into the RAM of the PITreader (see HTTP end point /api/transponder/userData/read (POST) [67]).

Request

- **Schematic representation**

  POST https://*<IP address>:<Port number>*/api/transponder/userData/write

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameters**

  None

- **Example**

  POST https://192.168.0.12/api/transponder/userData/write
  <empty / no body>

Response

General information can be found under Response format [10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [10].

Response in the event of a bad request with HTTP status code 400

- Content of the generic data field "msg": "" (empty string)
- Cause of error: transponder is not positioned

or

- Cause of error: User data on the PITreader was neither reset nor loaded from the transponder to the PITreader beforehand

Response in the event of a bad request with HTTP status code 500

- Content of the generic data field "msg": "" (empty string)
- Cause of error: error when writing the user data on the transponder. The created user data exceeds the transponder’s memory capacity.

5.4.9 HTTP end point /api/transponder/userData/clearGroup (POST)

Via the HTTP end point /api/transponder/userData/clearGroup, the parameter data for an individual device group can be deleted in the RAM of a PITreader using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/transponder/userData/clearGroup

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

deviceGroup <*Number*>	Device group to which the parameter belongs (Data type: NUMBER) Number: 0 … 9999

  Note: This data field is optional. If it is omitted, the "default values" are deleted.

- **Example**

```
POST https://192.168.0.12/api/transponder/userData/clearGroup
{
    "deviceGroup": 105
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible contents of the generic data field "msg":

- Cause of error: incorrect parameters

Content in the generic data field "msg":

– API.errorInvalidDevicegroupIndex: The value for "deviceGroup" is invalid.

5.4.10 HTTP end point /api/transponder/userData/addGroupValues (POST)

Via the HTTP end point /api/transponder/userData/addGroupValues, data can be added to a device group in the RAM of the PITreader using POST.

Request

- **Schematic representation**

  POST https://<IP address>:<Port number>/api/transponder/userData/addGroupValues

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

deviceGroup <*Number*>	Device group for which the value of the parameter is to be valid (optional) (Data type: NUMBER) Number: 0 … 9999 Note: The data field may not be transferred if the value of the parameter is to be valid for all device groups.
values	Transfer of the parameters (Data type: ARRAY of OBJECT)
id <*Number*>	Parameter ID (Data type: NUMBER) Number: 1 … 65535
type <*Type ID*>	Type ID of data type (Data type: NUMBER) Type ID: valid entry see [*1]
size <*Data length*>	Maximum number of characters for a parameter of the "STRING" data type (optional) (Data type: NUMBER) Data length: 2 … 255 (see [*1]) Note: "size" only needs to be transferred in the case of a parameter with type ID 1 (STRING). The number of characters (data length in Bytes) selected must be 1 Byte greater than the number of characters required; i.e.: size = <*Number of characters in Bytes*> + 1 Byte (= terminating character '\0') $size_{max}$ <= 255
numericValue <*Value*>	Numeric value of the parameter (optional) Value: valid entry depending on "type" (see [*1]) Note: The data field only needs to be transferred in the event of a parameter with a numeric value (type ID 10 … 15 or 30).
stringValue <*Value*>	String values of the parameter (optional) Value: valid entry depending on "type": Type ID 1: Valid UTF-8 string (max. number of characters = "size")

Type ID 20: String in accordance with RFC3339 section-5.6, date and time in UTC
- min. 01.01.2000 00:00 UTC
- Format: 2018-06-28T00:00:00Z
- The value "0" is translated into an empty string.

Note: The data field only needs to be transferred when a parameter with type ID 1 (STRING) or type ID 20 (DATETIME) is concerned.

Note: The POST request can only be processed correctly by the PITreader under the following conditions:
- The number of parameters transferred must not exceed the maximum possible number.
- The parameter ID of a transferred parameter must be available in the parameter definitions.
- The type ID of a numeric value must match the parameter definition and the numeric value must be within the value range.

The maximum number of characters ("size" data field) must match the parameter definition.

- **Example**

```
POST https://192.168.0.12/api/transponder/userData/addGroupValues
{
    "deviceGroup": 105,
    "values":
    [
        { "id": 10001, "type": 1, "size": 40, "stringValue": "James Pilz" }
    ]
}
```

[*1]

Note: The size of the JSON data in the body of requests using POST must not exceed 1100 Bytes. Requests with a larger data length must be split into several requests.

Type-ID	Name	Data length	Value range	Initial value
1	STRING	2 … 255 Byte		Empty STRING
10	INT8U	1 Byte	0 ... 255	0
11	INT8S	1 Byte	-128 ... 127	0
12	INT16U	2 Byte	0 ... 65535	0
13	INT16S	2 Byte	-32768 ... 32767	0
14	INT32U	4 Byte	0 ... 4294967295	0
15	INT32S	4 Byte	-2147483648 ... 2147483647	0
20	DATETIME	4 Byte		Empty time/date

Type-ID	Name	Data length	Value range	Initial value
30	PERMISSION	4 Byte	0 ... 64 (Hamming-coded)	0

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" can contain more precise information about the cause of the error.

Possible contents of the generic data field "msg":

- Cause of error: incorrect parameters

  Content in the generic data field "msg":

  – API.errorInvalidDeviceGroupIndex: The value for "deviceGroup" is invalid

  – API.errorEmptyData: "values" is not available or empty.

  – API.errorInvalidDataFormat: Invalid or unknown type ID,
  "type" does not match the parameter definition (data fields missing).

  – API.errorInvalidGroupValue:
  - The value for "numericValue" is invalid
  or
  - The string for "stringValue" is invalid. The maximum number of characters for "size" differs from the parameter definition.

5.5 External authentication

5.5.1 HTTP end point /api/status/authentication (POST)

Via the HTTP end point /api/status/authentication, permissions for a transponder can be defined in "External" mode using POST.

Request

- **Schematic representation**

  POST https://*<IP address>:<Port number>*/api/status/authentication

- **Required role number**

  ≥ 200 (transponder manager)

- **Parameter**

  The parameters must be transferred in the body of the request in JSON format.

securityId<*Security ID*>	Security ID, for which external authentication is to be defined (Data type: STRING)
permission <*Value*>	Permission that is to be defined for the stated transponder (Hamming coded, HD9). The permissions (Hamming codes) can be found under Overview of permissions [📖 92]. (Data type: NUMBER)
userData	Transfer of the parameters (Data type: ARRAY of OBJECT)
id <*Number*>	Parameter ID (Data type: NUMBER) Number: 1 … 65535
numericValue <*Value*>	Numeric value of the parameter (optional) Value: valid entry depending on the parameter's type ID (see [*1]) Note: The data field only needs to be transferred in the event of a parameter with a numeric value (type ID 10 … 15 or 30).
stringValue <*Value*>	String values of the parameter (optional) Value: valid entry depending on the parameter's type ID: Type ID 1: Valid UTF-8 string (max. number of characters depending on the parameter configuration) Type ID 20: String in accordance with RFC3339 section-5.6, date and time in UTC ▸ min. 01.01.2000 00:00 UTC ▸ Format: 2018-06-28T00:00:00Z ▸ The value "0" is translated into an empty string. Note: The data field only needs to be transferred when a parameter with type ID 1 (STRING) or type ID 20 (DATETIME) is concerned.

Note: The *securityID* parameter must be stated. In addition, at least one of the two parameters *permission* and *userData* must be stated.

Examples:

Request with the parameters "securityId" and "permission":

```
POST https://192.168.0.12/api/status/authentication
{
    "securityId": "309A68BACCA44C30",
    "permission": 116684
}
```

Request with the parameters "securityId", "permission" and "userData":

```
POST https://192.168.0.12/api/status/authentication
{
    "securityId": "309A68BACCA44C30",
    "permission": 116684
    "userData": [
                {
                    "id": 1000,
                    "numericValue": 20
                },
                {
                    "id": 1002,
                    "stringValue": "de-DE"
                }
    ]
}
```

[*1]

Type-ID	Name	Data length	Value range	Initial value
1	STRING	2 … 255 Byte		Empty STRING
10	INT8U	1 Byte	0 ... 255	0
11	INT8S	1 Byte	-128 ... 127	0
12	INT16U	2 Byte	0 ... 65535	0
13	INT16S	2 Byte	-32768 ... 32767	0
14	INT32U	4 Byte	0 ... 4294967295	0
15	INT32S	4 Byte	-2147483648 ... 2147483647	0
20	DATETIME	4 Byte		Empty time/date
30	PERMISSION	4 Byte	0 ... 64 (Hamming-coded)	0

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

Response in the event of a bad request with HTTP status code 400

In the event of a bad POST request at the end point api/status/authentication, the response from the web server may contain the HTTP status code 400 due to a variety of errors. In contrast to the content documented under Response format with HTTP status code 400 [📖 11], the generic data field "msg" contains more precise information about the cause of the error.

Possible contents of the generic data field "msg":

- Cause of error: incorrect parameters in the request

  Content in the generic data field "msg":

  – API.errorInvalidPermission: The value for "permission" is invalid.

  – API.errorEmptyData: The "userData" parameter is stated but does not contain any data.

- Potential error sources:

  – The value for "securityID" is invalid.

  – The transmitted security ID does not match the security ID of the current transponder.

  – There is no transponder in the read area.

  Content of the generic data field "msg":

  – API.errorInvalidSecurityId

- Potential error sources:

  – The device was not configured for "External" authentication mode and the "Allow external overwrite" option is not activated.

  – The "Allow external overwrite" option is not activated and the request contains the "userData" parameter.

  Content of the generic data field "msg":

  – API.errorInvalidState

- Potential error sources:

  The request contains the "userData" parameter and

  – The data format of the "id" parameter is incorrect (value is invalid).

  or

  – The data format of the "numericValue" parameter is incorrect (no number).

or

– The data format of the "stringValue" parameter is incorrect (no characters).

or

– The parameter ID does not exist in the user data configuration.

Content of the generic data field "msg":

– API.errorInvalidDataFormat

▸ Potential error sources:

The request contains the "userData" parameter and

– The value for "numericValue" is outside the valid value range.

or

– The parameter has type ID 1 and the string for "stringValue" exceeds the maximum number of characters.

or

– The parameter has type ID 20 and the string for "stringValue" is not a valid datum.

Content of the generic data field "msg":

– API.errorInvalidGroupValue

5.5.2 HTTP end point /api/led (GET)

Via the HTTP end point /api/led, the status of the LEDs and the "LED overwrite" settings can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/led

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/led

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
colour	NUMBER	Current LED colour Possible content:	
		0	Switched off
		1	Blue
		2	Yellow
		3	Red
		4	Green
flashMode	NUMBER	Current LED flash mode Possible content:	
		0	Static LED mode
		1	Flash mode (1Hz)
overwrite:	OBJECT	- - -	

Field name		Data type	Meaning	
	colour	NUMBER	LED colour for "LED overwrite" Possible content:	
			0	Switched off
			1	Blue
			2	Yellow
			3	Red
			4	Green
	flashMode	NUMBER	LED flash mode for "LED overwrite" Possible content:	
			0	Static LED mode
			1	Flash mode (1Hz)
	activated	BOOLEAN	LED overwrite status Possible content:	
			true	LED overwrite is activated
			false	LED overwrite is deactivated

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.5.3 HTTP end point /api/led (POST)

Via the HTTP end point /api/led, the "LED overwrite" settings can be defined using POST. With the help of the "LED overwrite" settings, LED colour and flash mode can be set externally. As a result, LED colour and flash mode no longer depend on the internal device status.

Request

Schematic representation

POST https://<IP address>:<Port number>/api/led

Required role number

≥ 200 (transponder manager)

Parameter

The parameters must be transferred in the body of the request in JSON format.

colour *<code>*	Colour code for "LED overwrite" (Data type: NUMBER): 0: switch off 1: blue 2: yellow 3: red 4: green Valid entry: 0 … 4
flashMode *<code>*	Flash code for "LED overwrite" (Data type: NUMBER): 0: static LED mode 1: Flash mode (1Hz) Valid entry: 0 or 1
activated *<true/false>*	true: Activate "LED overwrite" false: Deactivate "LED overwrite" Valid entry: true or false

Note: All the parameters transferred must be valid and at least one parameter must be transferred.

Example

```
POST https://192.168.0.12/api/led
{
    "colour": 2,
    "flashMode": 0,
    "activated": true
}
```

Response

General information can be found under Response format [📖 10].

In the event of a POST request, the response body will consist exclusively of the generic data fields.

Response in the event of HTTP status code 200

Information on the generic data fields in the event of an OK request can be found under Response format with HTTP status code 200 [📖 11].

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. Further information is available under Response format [📖 10].

5.6 "Single authentication" authentication type

5.6.1 HTTP end point /api/status/authentication/singleAuth (GET)

Via the HTTP end point /api/status/authentication/singleAuth, it is possible to use GET to enquire whether an authentication lock is set via the "Single authentication" authentication type and to enquire about details of the security ID of the locking transponder.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/status/authentication/singleAuth

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/status/authentication/singleAuth

Response

General information can be found under Response format [📖 10].

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format.

Field name	Data type	Meaning	
securityId	STRING	Security ID of the locking transponder Possible content:	
		"" (empty string)	No authentication lock is set.
		<Security ID>	Security ID of the locking transponder

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.7 Information on the current user

5.7.1 HTTP end point /api/me (GET)

Via the HTTP end point /api/me, information about the user who is currently signed in can be read using GET.

Request

- **Schematic representation**

  GET https://*<IP address>:<Port number>*/api/me

- **Required role number**

  ≥ 10 (guest)

- **Parameters**

  None

- **Example**

  GET https://192.168.0.12:443/api/me

Response

- Request-specific data fields in the body in the event of
  - Successful authentication
  - HTTP status code 200

  The response contains the following request-specific data fields and contents in the body in JSON format:

Field name	Data type	Meaning	
id	NUMBER	ID of the user who is currently logged in	
name	STRING	Name of the user who is currently logged in	
level	NUMBER	Access permission (role) of the user who is currently logged in	
type	NUMBER	User type Possible contents:	
		1	User
		2	API Client

Response in the event of a bad or forbidden request

INFORMATION

In the event of a bad or forbidden request, the response in the response header contains a corresponding HTTP status code. In such cases the response does not contain any request-specific data fields in the body. Further information is available under Response format [📖 10].

5.8 Status of the PITreader

5.8.1 HTTP end point /api/status/monitor (GET)

Via the HTTP end point /api/status/monitor, the status of the PITreader can be read and synchronisation monitoring can be restarted using GET.

Request

Schematic representation

GET https://*<IP address>*:*<Port number>*/api/status/monitor

Required role number

≥ 10 (guest)

INFORMATION

If the "syncStatus" parameter is used, a role number ≥ 200 (transponder manager) is required.

Parameters

The parameters must be transferred in the request as part of the URL.

syncStatus	Restart synchronisation monitoring (optional) (Data type: BOOLEAN)	
	false	Synchronisation monitoring is not restarted.
	true	Synchronisation monitoring is restarted. (Default value)

Note: If the "Synchronisation monitoring" option is active for the PITreader, synchronisation monitoring can be restarted with the syncStatus parameter. The time configured for the timeout is used.
If the "Synchronisation monitoring" option is not active and the syncStatus parameter is used, then the parameter is ignored. See also Synchronisation with external data [📖 14].

Example

POST https://192.168.0.12/api/status/monitor?syncStatus=true

Response

General information can be found under Response format [📖 10].

Request-specific data fields in the body in the event of

– Successful authentication

– HTTP status code 200

The response contains the following request-specific data fields and contents in the body in JSON format:

Field name		Data type	Meaning	
status:		OBJECT	- - -	
	fwVersion	STRING	Firmware version of the PITreader in the format "xxx.xxx.xxx"	
	ioPortValue	NUMBER	Signal at 24 V I/O port Possible content:	
			0	0 V
			1	24 V
	seuStatus	BOOLEAN	Connection to an evaluation unit SEU Possible content:	
			true	The PITreader is connected to an SEU and the connection is active.
			false	The PITreader is not connected to an SEU.
led:		OBJECT	- - -	
	colour	NUMBER	Current LED colour Possible content:	
			0	Switched off
			1	Blue
			2	Yellow
			3	Red
			4	Green
	flashMode	NUMBER	Current LED flash mode Possible content:	
			0	Static LED mode
			1	Flash mode (1 Hz)
config:		OBJECT	- - -	
	deviceGroup	NUMBER	Device’s device group for "transponder data" authentication mode	
	authenticationMode	NUMBER	Authentication mode Possible content:	
			0	External
			1	Transponder data
			2	Permission list
			3	Fixed permission

Field name		Data type	Meaning	
authentication		OBJECT	- - -	
	securityId	STRING	Transponder's security ID	
	authenticated	BOOLEAN	Transponder's authentication status Possible content:	
			true	The transponder was authenticated.
			false	The transponder was not authenticated.
	authenticationStatus	NUMBER	Status of authentication process Possible content:	
			0	No transponder
			1	Process completed
			2	Waiting for external authentication
	permission	NUMBER	Authenticated permission	
	failureReason	NUMBER	Reason for failed authentication Possible content:	
			0	No error
			1	No transponder positioned
			2	Permission "0" The transponder has no permissions for device groups.
			3	The validity of the transponder is outside the validity period. (Start date/end date)
			4	The transponder is included in the block list.
			5	No permission has been stored for the security ID yet. ("External" authentication mode)
			6	Authentication is locked by the 24 V I/O port.
			7	The "Single authentication" authentication type is configured and authentication is locked by another registered transponder.

Field name		Data type	Meaning	
			8	The "2-person rule" authentication type is configured and a second transponder is required for authentication.
			9	There is no permission in the permission list for the security ID. ("Permission list" authentication mode)
			10	Authentication is locked because a synchronisation timeout occurred.
	transponderUid	STRING	Transponder UID	
log:		OBJECT	- - -	
	latestEntry	OBJECT	Latest entry in the diagnostic log	
	▸ id	NUMBER	Diagnostic ID	
	▸ timestamp	STRING	Time stamp for the log entry (stated in UTC format) Example: 2018-06-28T00:39:53Z	
	▸ Index	NUMBER	Incremental index (Lamport clock)	
tags		OBJECT	Timer that is incremented each time an object changes (with some changes the counters are incremented multiple times). When the PITreader is restarted, all counters are set to a random value.	
	settings	NUMBER	Counters for changes to the device configuration	
	blockList	NUMBER	Counter for changes to the block list	
	permissionList	NUMBER	Counter for changes to the permission list	
	userDataConfig	NUMBER	Counter for changes to the user data	

6 Application guidelines

The following recommendations are intended to help you incorporate the PITreader into your application in the optimum way:

- Certificates

  Use a server certificate based on Elliptic Curve Cryptography (ECC). This corresponds to the PITreader's default settings.

- TLS Session Tickets

  We recommend you use a REST Client that supports TLS Session Tickets.

- Cyclical access to data

  We recommend you send a maximum of 2 requests per second to the PITreader's web server.

7 Overview of permissions

Permission	Code	
	Hexadecimal	Decimal
0	0x00000000	0
1	0x000001ff	511
2	0x00003e0f	15887
3	0x00003ff0	16368
4	0x0001c633	116275
5	0x0001c7cc	116684
6	0x0001f83c	129084
7	0x0001f9c3	129475
8	0x00064a55	412245
9	0x00064baa	412586
10	0x0006745a	423002
11	0x000675a5	423333
12	0x00078c66	494694
13	0x00078d99	495001
14	0x0007b269	504425
15	0x0007b396	504726
16	0x000a94aa	693418
17	0x000a9555	693589
18	0x000aaaa5	699045
19	0x000aab5a	699226
20	0x000b5299	742041
21	0x000b5366	742246
22	0x000b6c96	748694
23	0x000b6d69	748905
24	0x000cdeff	843519
25	0x000cdf00	843520
26	0x000ce0f0	844016
27	0x000ce10f	844047
28	0x000d18cc	858316
29	0x000d1933	858419
30	0x000d26c3	861891
31	0x000d273c	862012
32	0x00304c6a	3165290
33	0x00304d95	3165589

Permission	Code	
	Hexadecimal	Decimal
34	0x00307265	3175013
35	0x0030739a	3175322
36	0x00318a59	3246681
37	0x00318ba6	3247014
38	0x0031b456	3257430
39	0x0031b5a9	3257769
40	0x0036063f	3540543
41	0x003607c0	3540928
42	0x00363830	3553328
43	0x003639cf	3553743
44	0x0037c00c	3653644
45	0x0037c1f3	3654131
46	0x0037fe03	3669507
47	0x0037fffc	3670012
48	0x003ad8c0	3856576
49	0x003ad93f	3856703
50	0x003ae6cf	3860175
51	0x003ae730	3860272
52	0x003b1ef3	3874547
53	0x003b1f0c	3874572
54	0x003b20fc	3875068
55	0x003b2103	3875075
56	0x003c9295	3969685
57	0x003c936a	3969898
58	0x003cac9a	3976346
59	0x003cad65	3976549
60	0x003d54a6	4019366
61	0x003d5559	4019545
62	0x003d6aa9	4025001
63	0x003d6b56	4025174
64	0x00c04e98	12603032

8 HTTP status codes

The response to a request always contains an HTTP status code in the response header. The following list contains the possible HTTP status codes:

HTTP status code	Meaning
200	OK
400	Bad request
403	Forbidden
500	Internal server error

9 Time zones

The following time zones are available:

Name (web application)	Value (HTTP end point /api/config)
Pacific/Midway (-11:00)	Midway
Pacific/Honolulu (-10:00)	Honolulu
America/Anchorage (-9:00)	Anchorage
America/Inland Empire (-8:00)	Inland Empire
America/Los Angeles (-8:00)	Los Angeles
America/San Francisco (-8:00)	San Francisco
America/Tijuana (-8:00)	Tijuana
America/Denver (-7:00)	Denver
America/Phoenix (-7:00)	Phoenix
America/Chicago (-6:00)	Chicago
America/Dallas Fort Worth (-6:00)	Dallas Fort Worth
America/Guadalajara (-6:00)	Guadalajara
America/Houston (-6:00)	Houston
America/Mexico City (-6:00)	Mexico City
America/Regina (-6:00)	Regina
America/Atlanta (-5:00)	Atlanta
America/Boston (-5:00)	Boston
America/Miami (-5:00)	Miami
America/New York (-5:00)	New York
America/Philadelphia (-5:00)	Philadelphia
America/Washington, D.C. (-5:00)	Washington, D.C.
America/Halifax (-4:00)	Halifax
America/Manaus (-4:00)	Manaus
America/Santiago (-4:00)	Santiago
America/St Johns (-3:30)	St Johns
America/Belo Horizonte (-3:00)	Belo Horizonte
America/Buenos Aires (-3:00)	Buenos Aires
America/Rio de Janeiro (-3:00)	Rio de Janeiro
America/Sao Paulo (-3:00)	Sao Paulo
Europe/London (+0:00)	London
Africa/Brazzaville (+1:00)	Brazzaville
Europe/Barcelona (+1:00)	Barcelona
Europe/Belgrade (+1:00)	Belgrade
Europe/Berlin (+1:00)	Berlin

Name (web application)	Value (HTTP end point /api/config)
Europe/Hamburg (+1:00)	Hamburg
Europe/Madrid (+1:00)	Madrid
Europe/Milan (+1:00)	Milan
Europe/Munich (+1:00)	Munich
Europe/Rhein-Ruhr (+1:00)	Rhine-Ruhr
Europe/Rome (+1:00)	Rome
Europe/Sarajevo (+1:00)	Sarajevo
Europe/Stuttgart (+1:00)	Stuttgart
Africa/Cairo (+2:00)	Cairo
Africa/Harare (+2:00)	Harare
Asia/Jerusalem (+2:00)	Jerusalem
Europe/Helsinki (+2:00)	Helsinki
Africa/Nairobi (+3:00)	Nairobi
Asia/Kuwait (+3:00)	Kuwait
Europe/Ankara (+3:00)	Ankara
Europe/Istanbul (+3:00)	Istanbul
Europe/Moscow (+3:00)	Moscow
Asia/Tehran (+3:30)	Tehran
Asia/Dubai (+4:00)	Dubai
Asia/Karachi (+5:00)	Karachi
Asia/Lahore (+5:00)	Lahore
Asia/Ahmedabad (+5:30)	Ahmedabad
Asia/Bangalore (+5:30)	Bangalore
Asia/Calcutta (+5:30)	Calcutta
Asia/Chennai (+5:30)	Chennai
Asia/Delhi (+5:30)	Delhi
Asia/Hyderabad (+5:30)	Hyderabad
Asia/Kolkata (+5:30)	Kolkata
Asia/Mumbai (+5:30)	Mumbai
Asia/Pune (+5:30)	Pune
Asia/Surat (+5:30)	Surat
Asia/Kathmandu (+5:45)	Kathmandu
Asia/Rangoon (+6:30)	Rangoon
Asia/Bangkok (+7:00)	Bangkok
Asia/Beijing (+8:00)	Beijing
Asia/Changzhou (+8:00)	Changzhou

Name (web application)	Value (HTTP end point /api/config)
Asia/Chengdu (+8:00)	Chengdu
Asia/Chongqing (+8:00)	Chongqing
Asia/Guangzhou (+8:00)	Guangzhou
Asia/Hangzhou (+8:00)	Hangzhou
Asia/Hong Kong (+8:00)	Hong Kong
Asia/Jinan (+8:00)	Jinan
Asia/Nanchang (+8:00)	Nanchang
Asia/Nanjing (+8:00)	Nanjing
Asia/Qingdao (+8:00)	Qingdao
Asia/Shanghai (+8:00)	Shanghai
Asia/Shantou (+8:00)	Shantou
Asia/Shenyang (+8:00)	Shenyang
Asia/Shenzhen (+8:00)	Shenzhen
Asia/Taipei (+8:00)	Taipei
Asia/Tianjin (+8:00)	Tianjin
Asia/Wenzhou (+8:00)	Wenzhou
Asia/Wuhan (+8:00)	Wuhan
Asia/Xi'an (+8:00)	Xi'an
Asia/Zhengzhou (+8:00)	Zhengzhou
Australia/Perth (+8:00)	Perth
Asia/Nagoya (+9:00)	Nagoya
Asia/Osaka (+9:00)	Osaka
Asia/Seoul (+9:00)	Seoul
Asia/Tokyo (+9:00)	Tokyo
Australia/Adelaide (+9:30)	Adelaide
Australia/Darwin (+9:30)	Darwin
Australia/Brisbane (+10:00)	Brisbane
Australia/Hobart (+10:00)	Hobart
Australia/Sydney (+10:00)	Sydney
Pacific/Guam (+10:00)	Guam
Pacific/Auckland (+12:00)	Auckland

Support

Technical support is available from Pilz round the clock.

Americas
Brazil
+55 11 97569-2804
Canada
+1 888 315 7459
Mexico
+52 55 5572 1300
USA (toll-free)
+1 877-PILZUSA (745-9872)

Asia
China
+86 21 60880878-216
Japan
+81 45 471-2281
South Korea
+82 31 778 3300

Australia and Oceania
Australia
+61 3 95600621
New Zealand
+64 9 6345350

Europe
Austria
+43 1 7986263-0
Belgium, Luxembourg
+32 9 3217570
France
+33 3 88104003
Germany
+49 711 3409-444
Ireland
+353 21 4804983
Italy, Malta
+39 0362 1826711
Scandinavia
+45 74436332
Spain
+34 938497433
Switzerland
+41 62 88979-32
The Netherlands
+31 347 320477
Turkey
+90 216 5775552
United Kingdom
+44 1536 462203

You can reach our international hotline on:
+49 711 3409-222
support@pilz.com

Pilz develops environmentally-friendly products using ecological materials and energy-saving technologies. Offices and production facilities are ecologically designed, environmentally-aware and energy-saving. So Pilz offers sustainability, plus the security of using energy-efficient products and environmentally-friendly solutions.

BLUEcoMPETENCE Alliance Member
Partner of the Engineering Industry Sustainability Initiative

BRANDS OF THE CENTURY PILZ 2019 GERMAN STANDARDS

Quality Management ISO 9001:2015 REG. NO. 2462
Environmental Management ISO 14001 REG. NO. 2462
Energy Management ISO 50001 REG. NO. 2462

1005365-EN-06, 2023-03 Printed in Germany

We are represented internationally. Please refer to our homepage www.pilz.com for further details or contact our headquarters.

Headquarters: Pilz GmbH & Co. KG, Felix-Wankel-Straße 2, 73760 Ostfildern, Germany
Telephone: +49 711 3409-0, Telefax: +49 711 3409-133, E-Mail: info@pilz.com, Internet: www.pilz.com

PILZ
THE SPIRIT OF SAFETY