



ibaPDA-Interface-HTTP(S)

Data Interface for HTTP(S)

Manual

Issue 1.0

Measurement Systems for Industry and Energy

www.iba-ag.com

Manufacturer

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Contacts

Main office +49 911 97282-0
Support +49 911 97282-14
Engineering +49 911 97282-13
E-mail iba@iba-ag.com
Web www.iba-ag.com

Unless explicitly stated to the contrary, it is not permitted to pass on or copy this document, nor to make use of its contents or disclose its contents. Infringements are liable for compensation.

© iba AG 2024, All rights reserved.

The content of this publication has been checked for compliance with the described hardware and software. Nevertheless, discrepancies cannot be ruled out, and we do not provide guarantee for complete conformity. However, the information furnished in this publication is updated regularly. Required corrections are contained in the following regulations or can be downloaded on the Internet.

The current version is available for download on our web site www.iba-ag.com.

Version	Date	Revision	Author	Version SW
1.0	07- 2024	First issue	st	8.7.0

Windows® is a brand and registered trademark of Microsoft Corporation. Other product and company names mentioned in this manual can be labels or registered trademarks of the corresponding owners.

Contents

1	About this documentation	4
1.1	Target group and previous knowledge	4
1.2	Notations	4
1.3	Used symbols.....	5
2	System requirements	6
3	The HTTP(S) interface.....	7
3.1	System topologies.....	7
3.2	Configuration and engineering ibaPDA.....	8
3.2.1	Interface settings	8
3.2.2	Adding a module.....	9
3.2.3	General module settings.....	9
3.2.4	Request settings.....	11
3.2.5	Response tab	19
3.2.6	Analog and digital signals	23
4	Application example HTTP(S)	24
4.1	Preparation	24
4.2	Configuration of the ibaPDA HTTP(S) module	25
4.3	Further processing of the requested data	28
4.4	Configuring the text splitter module	29
5	Diagnostics.....	31
5.1	License	31
5.2	Log files.....	31
5.3	Error messages	32
5.4	Connection diagnostics with PING.....	33
5.5	Diagnostic modules	34
6	Support and contact.....	39

1 About this documentation

This documentation describes the function and application of the software interface *ibaPDA-Interface-HTTP(S)*.

This documentation is a supplement to the *ibaPDA* manual. Information about all the other characteristics and functions of *ibaPDA* can be found in the *ibaPDA* manual or in the online help.

1.1 Target group and previous knowledge

This manual is aimed at qualified professionals who are familiar with handling electrical and electronic modules as well as communication and measurement technology. A person is regarded as professional if he/she is capable of assessing the work assigned to him/her and recognizing possible risks on the basis of his/her specialist training, knowledge and experience and knowledge of the standard regulations.

For the handling of *ibaPDA-Interface-HTTP(S)* the following basic knowledge is required and/or useful:

- Windows operating system
- Basic knowledge of *ibaPDA*
- Basic knowledge of the HTTP(S) protocol

1.2 Notations

In this manual, the following notations are used:

Action	Notation
Menu command	Menu <i>Logic diagram</i>
Calling the menu command	<i>Step 1 – Step 2 – Step 3 – Step x</i> Example: Select the menu <i>Logic diagram – Add – New function block</i> .
Keys	<Key name> Example: <Alt>; <F1>
Press the keys simultaneously	<Key name> + <Key name> Example: <Alt> + <Ctrl>
Buttons	<Key name> Example: <OK>; <Cancel>
Filenames, paths	Filename, Path Example: Test.docx

1.3 Used symbols

If safety instructions or other notes are used in this manual, they mean:

Danger!



The non-observance of this safety information may result in an imminent risk of death or severe injury:

- Observe the specified measures.
-

Warning!



The non-observance of this safety information may result in a potential risk of death or severe injury!

- Observe the specified measures.
-

Caution!



The non-observance of this safety information may result in a potential risk of injury or material damage!

- Observe the specified measures
-

Note



A note specifies special requirements or actions to be observed.

Tip



Tip or example as a helpful note or insider tip to make the work a little bit easier.

Other documentation



Reference to additional documentation or further reading.

2 System requirements

The following system requirements are necessary for the use of the HTTP(S) data interface:

- *ibaPDA* v8.7.0 or higher
- License for *ibaPDA-Interface-HTTP(S)*
- Internet connection

For more requirements on the PC hardware used and the supported operating systems, please refer to the *ibaPDA* documentation.

Order no.	Product name	Description
31.001018	ibaPDA-Interface-HTTP(S)	Extension license for an ibaPDA system with the data interface HTTP(S), no limitation of connections

3 The HTTP(S) interface

With the data interface *ibaPDA-Interface-HTTP(S)*, *ibaPDA* offers the possibility to configure requests via HTTP(S) and to receive data via HTTP(S). Users can request values from web services via HTTP(S) and write them there. The result can be processed further in a text signal.

These HTTP methods are supported: Get, Post, Put, Delete, Patch, Options, Trace, Head.

The requests can be executed periodically or on trigger.

3.1 System topologies

The connections to the web servers can be established via the computer's standard Ethernet interface.

No further software is necessary for operation.

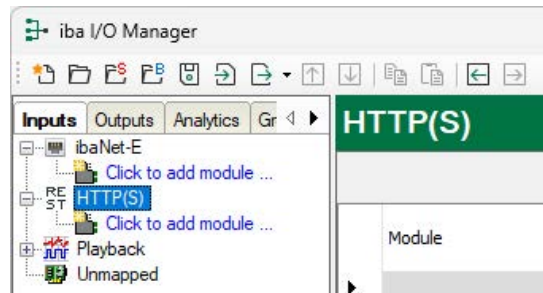
Note



It is recommended carrying out the TCP/IP communication on a separate network segment to exclude a mutual influence by other network components.

3.2 Configuration and engineering ibaPDA

The engineering for *ibaPDA* is described in the following. If all system requirements are fulfilled, *ibaPDA* displays the *HTTP(S)* interface in the signal tree of the I/O Manager.



Note



An application example for an HTTP(S) request can be found in chapter [Application example HTTP\(S\)](#), page 24. The example uses a request formulated in java script to explain the configuration of the HTTP(S) module as well as the further processing and display of the requested data.

3.2.1 Interface settings

The interface overview contains a list of the configured HTTP(S) modules with statistical information for each module.

 A screenshot of the 'iba I/O Manager' software window. The 'HTTP(S)' module is selected, and a table with statistical information is displayed. The table has columns for 'Module', 'Read counter', 'Error counter', 'Data size', and 'Response time' (subdivided into 'Actual', 'Min', and 'Max'). The data rows contain question marks, indicating that no data has been recorded yet.

Module	Read counter	Error counter	Data size	Response time		
				Actual	Min	Max
?	?	?	?	?	?	?
?	?	?	?	?	?	?
?	?	?	?	?	?	?

Connection table

The table shows the configured connections. Each table row corresponds to an HTTP(S) module or a connection.

<Open log file>

If HTTP(S) connections have been established, all connection-specific operations are logged in a text file. Using this button, you can open and see this file. In the file system on the hard disk, you will find the log file in the program path of ibaPDA (C:\ProgramData\iba\ibaPDA\Log). The file name of the current log file is [httpLog.txt](#), the name of the archived log files is [httpLog_yyy_mm_dd_hh_mm_ss.txt](#).

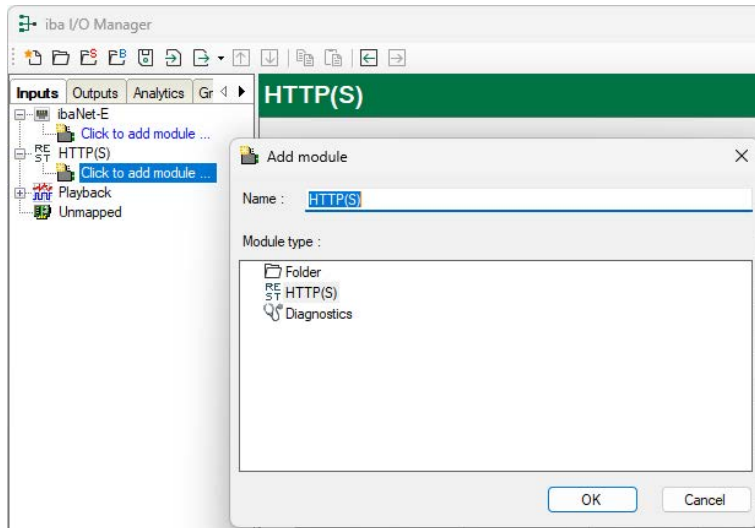
<Reset statistics>

If you would like to reset the counters for all connections, click on the <Reset statistics> button.

3.2.2 Adding a module

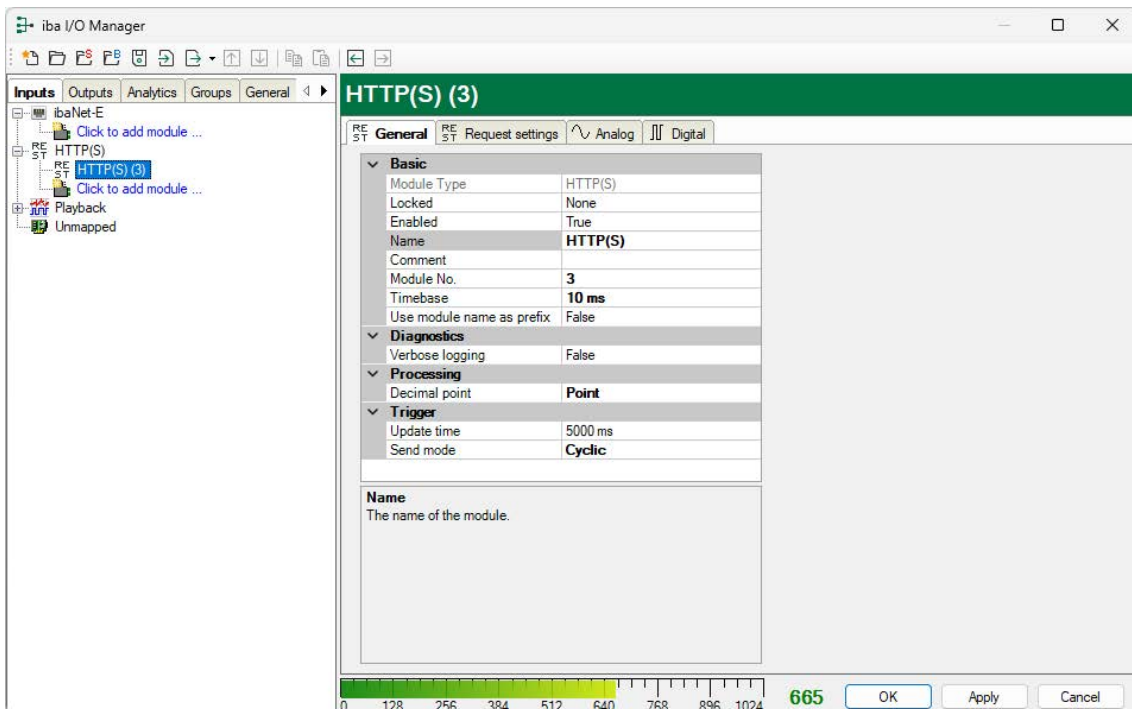
An HTTP(S) module is required for each connection. You can also add one or more diagnostic modules. For more information on the diagnostic modules, refer to chapter ↗ *Diagnostic modules*, page 34.

1. Add a module by clicking on the blue command *Click to add module* below the interface.
2. Select the HTTP(S) module type, assign a name if required and click <OK>.



3.2.3 General module settings

The following module settings can be configured on the *General* tab:



Basic settings

Module Type (information only)

Indicates the type of the current module.

Locked

You can lock a module to avoid unintentional or unauthorized changing of the module settings.

Enabled

Enable the module to record signals.

Name

You can enter a name for the module here.

Comment

You can enter a comment or description of the module here. This will be displayed as a tooltip in the signal tree.

Module No.

This internal reference number of the module determines the order of the modules in the signal tree of *ibaPDA* client and *ibaAnalyzer*.

Timebase

All signals of the module are sampled on this timebase.

Use module name as prefix

This option puts the module name in front of the signal names.

Diagnostics

Verbose logging

If verbose logging is disabled (False), only errors are logged. If this is enabled (True), all request and response details are logged.

Trigger

Update time

Depending on the send mode, the update time defines the delay between two consecutive requests.

- *Cyclic*: A request is sent periodically based on the update time.
- *On change/on trigger*: The update time acts as a debounce time. If less time has elapsed between the last request and the new trigger than was defined in *Update time*, the request is ignored.

Send mode

This determines when a new HTTP(S) request will be sent.

- *Cyclic*: A request is sent periodically based on the update time.
- *On change*: A request is sent each time the data of the trigger signal changes.
- *On trigger*: A request is sent each time a rising edge is detected on the trigger signal.

Trigger signal

The *Trigger signal* field appears if the send mode *On change* or *On trigger* is selected. The trigger signal determines when a new HTTP(S) request will be sent.

You can select an analog, digital or text signal in send mode *On change*. You can select a digital signal in send mode *On trigger*.

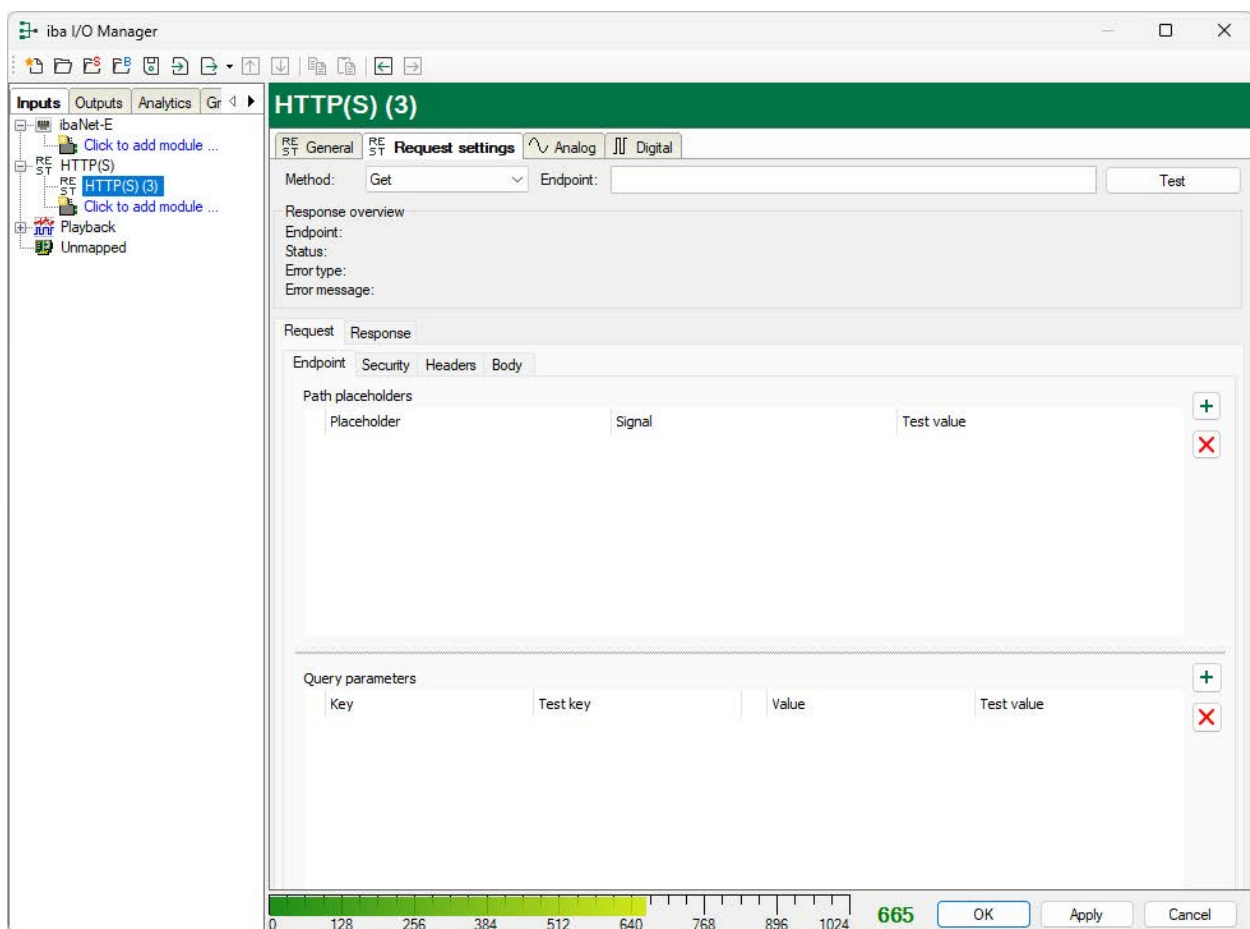
Processing

Decimal point

Configure which character is used for the decimal point.

3.2.4 Request settings

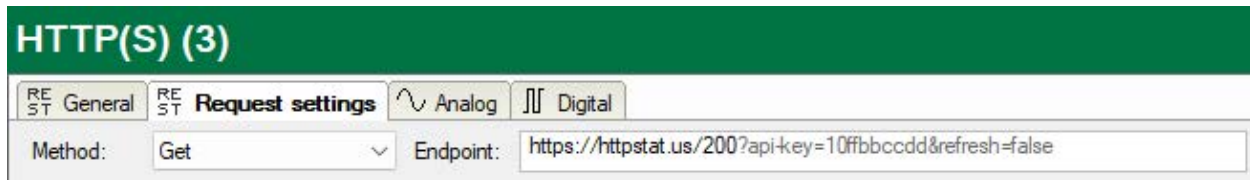
Configure the settings for a request in the *Request settings* tab.



Endpoint

Define the URL for the request in the *Endpoint* text field. Select the HTTP method in the *Method* selection list. The HTTP(S) module supports all standard HTTP methods.

An endpoint consists of the path and the query, separated by "?".

**Example:**

GET https://httpstat.us/200?api-key=10ffbcccdd&refresh=false

Method: GET

Path: https://httpstat.us/200

Request: api-key=10ffbcccdd&refresh=false

3.2.4.1 Request – Endpoint tab

When defining the endpoint, you can use query parameters and path placeholders. The *Endpoint* tab lists the query parameters and defines the path placeholders.

An endpoint can be pasted here directly using copy & paste. The parameters beginning with the character string `?` are automatically created as query parameters.

The character string `?` can not be entered manually. A query parameter must be created for this.

3.2.4.1.1 Request settings – query parameter

The query parameters are listed in a grid. A query parameter always consists of a key and a value.

You can add a new query parameter using the `<+>` button next to the grid. The `<x>` button deletes a selected query parameter.

Following the example of `https://httpstat.us/200?api-key=10ffbcccdd&refresh=false`, the query parameters grid contains two key-value pairs with four columns.

Key	Test key	Value	Test value
✓ api-key		✓ 10ffbcccdd	
✓ refresh		✓ false	

Key

The key of the query parameter can be a static value or a signal.

Test key

If a signal is selected for key, the value in the *Test key* column is used as the key for the query parameter when testing the request. Otherwise the column is not visible.

Value

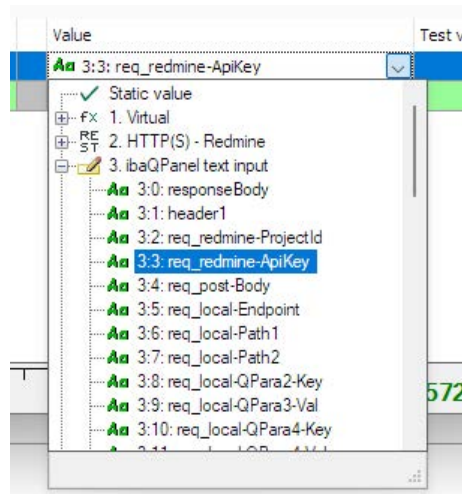
The value of the query parameter can be a static value or a signal.

Test value

If a signal is selected for the value, the value in the *Test value* column is used as the value for the query parameter when testing the request. Otherwise the column is not visible.

If the values in the grid are changed, the endpoint text field is updated automatically and vice versa.

In the following example, a signal is selected as value for the query parameter.



The complete endpoint of the current configuration is as follows:

[https://httpstat.us/200?api-key={{\[3:3\]}}&refresh=false](https://httpstat.us/200?api-key={{[3:3]}}&refresh=false)

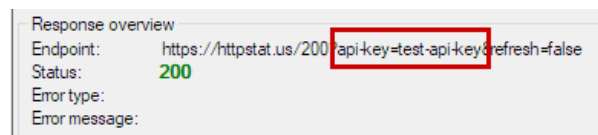
If the request is sent while the acquisition is running, the placeholder `{{[3:3]}}` will be replaced with the current value of the signal.

For example, assuming signal `[3:3]` currently has the value "test", the endpoint will be resolved to

<https://httpstat.us/200?api-key=test&refresh=false>.

However, if you test the request, the value of the signal is not available. You can therefore define a value for the test in the *Test value* column.

After using the <Test> button, the response overview shows that the endpoint has resolved the value of the api-key parameter `{{[3:3]}}` to the defined test value (test-api-key).



3.2.4.1.2 Request settings – Path placeholder

Path placeholders are used to dynamically change a part of the path. In these cases, no static value is sent, but the current value of the defined signal. Static values have to be defined directly in the endpoint text field.

HTTP(S) (3)

RE ST General RE ST **Request settings** Analog Digital

Method: Get Endpoint: https://httpstat.us/200?api-key=10ffbccdd&refresh=false&key3=value3 Test

Response overview
Endpoint:
Status:
Error type:
Error message:

Request Response

Endpoint Security Headers Body

Path placeholders

Placeholder	Signal	Test value

Query parameters

Key	Value
api-key	10ffbccdd
refresh	false
key3	value3

You can add a new path placeholder using the <+> button next to the grid. The <x> button deletes a selected path placeholder.

A path placeholder consists of the placeholder, the signal and the test value.

Placeholder

The placeholder is generated automatically. The designation can be changed if required. The name for the placeholder must be unique and start and end with a curly bracket.

Signal

Select the signal whose value replaces the placeholder here.

Test value

Define a test value here that is used when testing the request.

In the following example, the signal [3:6] is selected for the placeholder {1} and the test value is defined as 200.

Placeholder	Signal	Test value
{1}	3:6: req_local-Path1	200

The *Endpoint* text field is automatically updated according to the example.

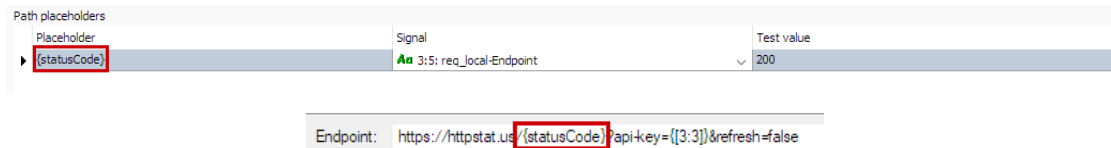
Method: Get Endpoint: https://httpstat.us/{1}?api-key={3:3}&refresh=false

When the request is sent, {1} is replaced by the test value 200.



You can also use a detailed designation instead of {1}. (e. g. {statusCode}).

Example:



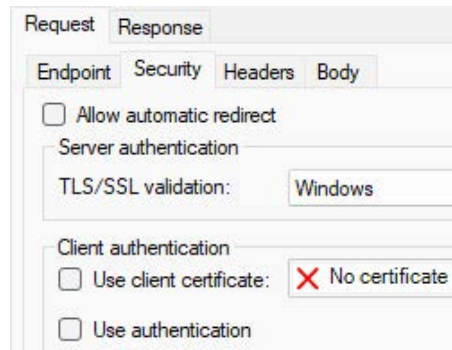
Note



A placeholder can be moved in the *Endpoint* text field. To do this, select the placeholder, cut it with <CTRL> + <X> and paste it elsewhere with <CTRL> + <V>.

3.2.4.2 Request – Security tab

The *Security* tab provides various security related settings.



Allow automatic redirect

If the destination responds with a redirect (HTTP status code 300 to 399), the redirection is followed if this option is enabled.

Server authentication

TLS/SSL validation

When connecting to an HTTPS endpoint, the SSL certificates are checked. Select the behavior for certificate validation here.

- *None*: The SSL certificate is not validated. This option enables insecure connections and is not recommended.
- *Windows*: The SSL certificate is automatically validated by the operating system.
- *Certificate store*: The SSL certificate is validated by the *ibaPDA* certificate store.

Client authentication

Client certificate

If this option is enabled, the certificate selected here is sent as a client certificate with the request.

Use authentication

If this option is enabled, the request is sent with the configured authentication.

ibaPDA supports different types of authentication:

- Basic
- Json Web Token
- OAuth2.0 (ROPC and client credentials)

Select the desired authentication type from the *Type* drop-down list. The following configuration options change accordingly.

Basic

With basic authentication, you can define a user name and password. The authentication is added as the Authorization header to the request.

The screenshot shows two panels. The left panel, titled 'Settings', contains three fields: 'Type' with a dropdown menu set to 'Basic', 'Add to' with a dropdown menu set to 'Header', and 'Query parameter' with an empty text box. The right panel, titled 'Basic', contains two fields: 'Username' with the text 'testUser' and 'Password' with a masked field of seven asterisks.

JSON Web Token (JWT)

Select the type for JWT authentication:

- Header: The JSON web token is sent in the Authorization header as a bearer token.
- Query: The JSON web token is sent as a query parameter in the URL with the provided key. Enter the query parameter as well.

The screenshot shows two panels. The top panel, titled 'Settings', contains three fields: 'Type' with a dropdown menu set to 'Json Web Token (JWT)', 'Add to' with a dropdown menu set to 'Query', and 'Query parameter' with an empty text box. The bottom panel, titled 'Json Web Token (JWT)', contains three fields: 'Algorithm' with a dropdown menu set to 'HS256', 'Secret' with an empty text box, and 'Payload' with a large empty text area.

Additional parameters for both types:

Algorithm

The algorithm is used to generate the signature. Currently only HS256 (HMAC SHA256) is supported.

Secret

The secret is used as the key for the HMAC SHA256 algorithm.

Payload

The payload contains custom information used to validate the request.

OAuth2.0

The OAuth2.0 protocol defines multiple authorization standards. *ibaPDA* supports the ROPC (username/password) and the *client credentials* grant.

“Three-legged” grants are currently not supported.

Note

Requests are automatically authorized using the given information. Subsequent requests will use the issued token. If the token is expired and a refresh token is available, the token will be refreshed with the provided information.

Select the desired grant type from the drop-down list:

- Password
- Client credentials

Some common properties apply to both grant types.

Access token URL

Defines the endpoint for getting and refreshing the token.

Scope

Optionally defines the requested scope of the token. A space-separated list of scopes or authorizations required by *ibaPDA*.

Client ID

The application ID (client ID) that is assigned to *ibaPDA*.

Client secret

Optionally defines the client secret.

Send client authentication in

Defines whether the authentication for token-related requests is added to the header or the body.

Grant type password:

Besides the common properties, the *password* grant type supports adding a username and a password.

The screenshot shows the configuration for the OAuth 2.0 Password grant type. On the left, under 'Settings', the 'Type' is set to 'OAuth 2.0' and 'Add to' is set to 'Header'. The main configuration area on the right is titled 'OAuth 2.0' and 'Grant type: Password'. It includes the following fields: 'Access token url' (https://api.test.com/oauth/token), 'Scope' (read_user), 'Client ID' (110ddf), 'Client secret' (*****), 'Send client authentication in' (radio buttons for Header and Body, with Body selected), 'Username' (testUser), and 'Password' (*****).

Grant type client credentials

The common properties apply here, without password and user name.

The screenshot shows the configuration for the OAuth 2.0 Client credentials grant type. On the left, under 'Settings', the 'Type' is set to 'OAuth 2.0' and 'Add to' is set to 'Header'. The main configuration area on the right is titled 'OAuth 2.0' and 'Grant type: Client credentials'. It includes the following fields: 'Access token url' (https://api.test.com/oauth/token), 'Scope' (read_user), 'Client ID' (100ddf), 'Client secret' (*****), and 'Send client authentication in' (radio buttons for Header and Body, with Body selected).

3.2.4.3 Request – Header tab

The request headers are defined in the *Header* tab.

The screenshot shows the 'Request Headers' configuration interface. It has tabs for 'Request' and 'Response', and sub-tabs for 'Endpoint', 'Security', 'Headers', and 'Body'. The 'Request Headers' section contains a table with columns for 'Key', 'Test key', 'Value', and 'Test value'. There are '+' and '-' buttons on the right side of the table to add and delete headers.

Key	Test key	Value	Test value

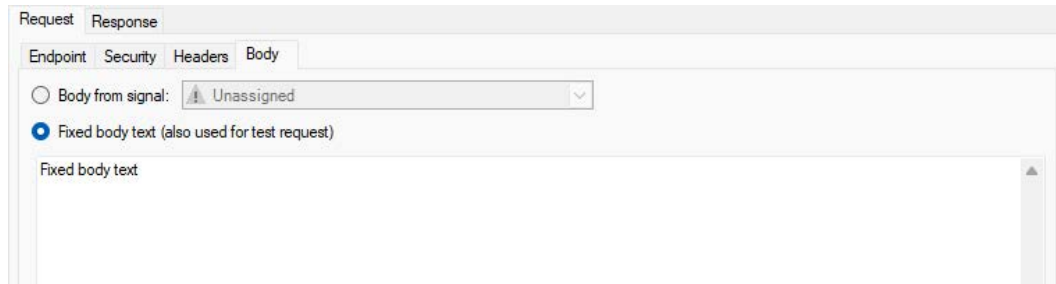
A static value can be entered or a signal can be selected for the key and the value.

Use the <+> button to add a new request header and the <x> button to delete a selected request header.

3.2.4.4 Request – Body tab

The body of an HTTP request contains all the information that is sent to the web server. This can either be a fixed text or a signal. Select the desired option.

If you have selected *Fixed body text*, you can enter the text in the text field below.



Note

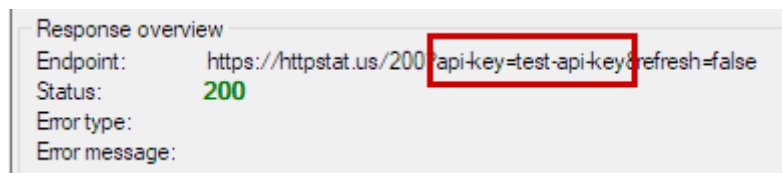


Even if the *Body from signal* option is selected, the text input field remains active, as the text in the text field can be used as a test value to test the request.

3.2.5 Response tab

The settings in the *Response* tab relate to the response that follows a request.

After testing a request, the response overview provides an overview of the response. The overview contains the resolved endpoint, the status code, the error type and the error message.



If the request fails, an error message appears. The error can then be identified by the type of error and the error message.

Depending on the selected authentication, the response overview also shows errors that occurred during the authentication process.

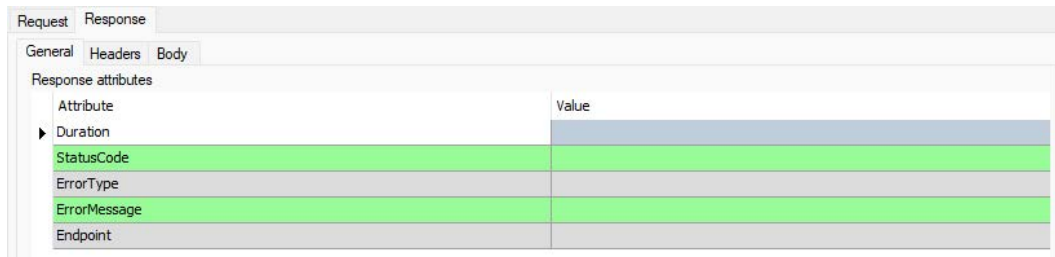
Possible error messages:

Error type	Cause:
RequestFailed	The request failed. No further information available.
SslVerificationFailed	An error occurred while establishing an SSL connection.
StatusCodeNotSuccessful	The HTTP status code is not successful (4xx).
Timeout	The server did not respond in time.
InvalidEndpoint	The URL is not well formatted.
IllegalPlaceholderCharacter	The value of a placeholder contains illegal characters.

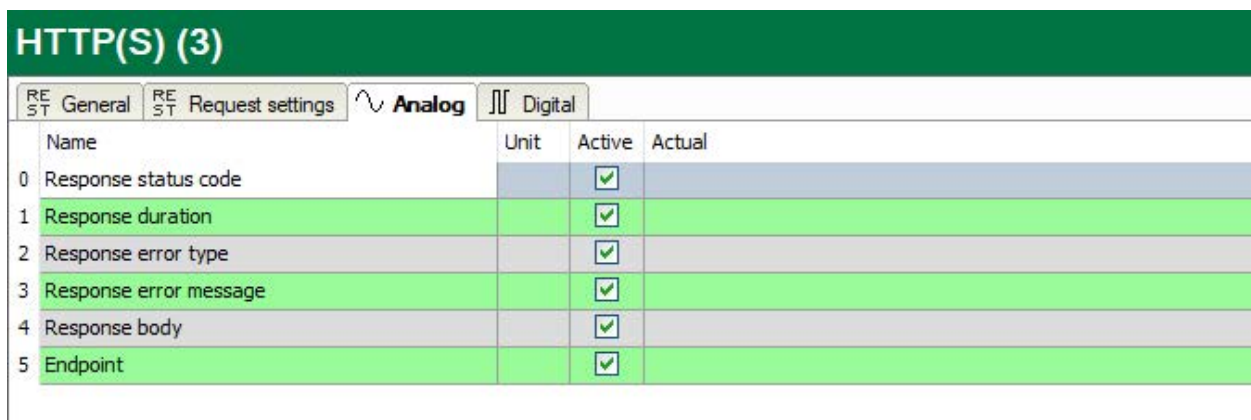
Error type	Cause:
OAuth2MissingAuthorizationEndpoint	Only applies to OAuth2: The access token URL is missing.
OAuth2InvalidAuthorizationResponse	Only applies to OAuth2: The request sent to the authorization server was not successful.
OAuth2PasswordUsernameMissing	Only applies to OAuth2: ROPC The username or password is missing.
OAuth2InvalidRequest	Only applies to OAuth2: The request is invalid. For more details see https://datatracker.ietf.org/doc/html/rfc6749#section-5.2 ("invalid_request")
OAuth2InvalidClient	Only applies to OAuth2: The client authentication failed. For more details see https://datatracker.ietf.org/doc/html/rfc6749#section-5.2 ("invalid_client")
OAuth2InvalidGrant	Only applies to OAuth2: The authorization grant is invalid. For more details see https://datatracker.ietf.org/doc/html/rfc6749#section-5.2 ("invalid_grant")
OAuth2UnauthorizedClient	Only applies to OAuth2: The client is not authorized for requesting the grant. For more details see https://datatracker.ietf.org/doc/html/rfc6749#section-5.2 ("unauthorized_client")
OAuth2UnsupportedGrantType	Only applies to OAuth2: The requested grant is not supported. For more details see https://datatracker.ietf.org/doc/html/rfc6749#section-5.2 ("unsupported_grant")
OAuth2InvalidScope	Only applies to OAuth2: The request scope is not valid. For more details see https://datatracker.ietf.org/doc/html/rfc6749#section-5.2 ("invalid_scope")
InvalidClientCertificate	The selected client certificate is not valid..
ServerCertificateInvalidDate	The date of the SSL certificate is not in a valid range.
ServerCertificateNotTrusted	The SSL certificate exists in the certificate store but is not trusted.
ServerCertificateInvalid	The SSL certificate is invalid.
ServerCertificateChainNotTrusted	The chain of the SSL certificate is not trusted.
ServerCertificateChainNotValid	The chain of the SSL certificate is not valid.
ServerCertificateMissingInManager	The SSL certificate is missing in the certificate store.

3.2.5.1 Response – General tab

The response attributes contain information about the response, such as the duration, status code, error type, error message and endpoint.

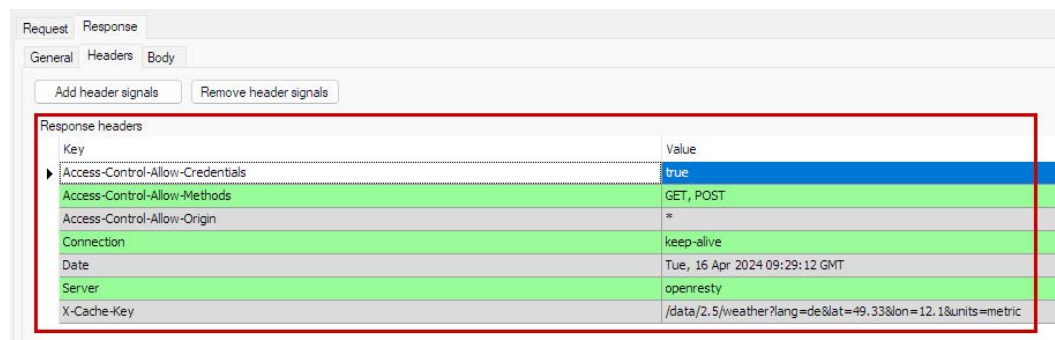


The HTTP(S) module also contains auto-generated signals for those attributes. The signals can be found in the *Analog* tab of the module.



3.2.5.2 Response – Header tab

The response headers grid contains the headers received in the response.



Using the <Add header signals> and the <Remove header signals> buttons, the headers can either be added to or removed from the signals of the module. Once the signals have been added, they appear in the *Analog* tab.

HTTP(S) - Weather (6)			
RE ST	General	RE ST Request settings	Analog
Name	Unit	Active	A
0	Response status code	<input checked="" type="checkbox"/>	
1	Response duration	<input checked="" type="checkbox"/>	
2	Response error type	<input checked="" type="checkbox"/>	
3	Response error message	<input checked="" type="checkbox"/>	
4	Response body	<input checked="" type="checkbox"/>	
5	Connection	<input checked="" type="checkbox"/>	
6	Access-Control-Allow-Credentials	<input checked="" type="checkbox"/>	
7	Access-Control-Allow-Methods	<input checked="" type="checkbox"/>	
8	Access-Control-Allow-Origin	<input checked="" type="checkbox"/>	
9	Connection	<input checked="" type="checkbox"/>	
10	Date	<input checked="" type="checkbox"/>	
11	Server	<input checked="" type="checkbox"/>	
12	X-Cache-Key	<input checked="" type="checkbox"/>	

3.2.5.3 Response – Body tab

The data received is displayed in the *Response - Body* tab.

Request	Response
General	Headers
Body content:	
<pre> { "coord": { }, "weather": [{ "id": 804, "main": "Clouds", "description": "Bedeckt", "icon": "04d" }], "base": "stations", "main": { "temp": 26.62, "feels_like": 26.62, "temp_min": 25.01, "temp_max": 28.62, "pressure": 1010, "humidity": 34, "sea_level": 1010, "gmd_level": 962 }, "visibility": 10000 } </pre>	

The received data is also available as an automatically generated signal *Response body* in the *Analog* tab.

HTTP(S) - Weather (6)			
RE ST	General	RE ST Request settings	Analog
Name	Unit	Active	A
0	Response status code	<input checked="" type="checkbox"/>	
1	Response duration	<input checked="" type="checkbox"/>	
2	Response error type	<input checked="" type="checkbox"/>	
3	Response error message	<input checked="" type="checkbox"/>	
4	Response body	<input checked="" type="checkbox"/>	
5	Connection	<input checked="" type="checkbox"/>	
6	Access-Control-Allow-Credentials	<input checked="" type="checkbox"/>	
7	Access-Control-Allow-Methods	<input checked="" type="checkbox"/>	
8	Access-Control-Allow-Origin	<input checked="" type="checkbox"/>	
9	Connection	<input checked="" type="checkbox"/>	
10	Date	<input checked="" type="checkbox"/>	
11	Server	<input checked="" type="checkbox"/>	
12	X-Cache-Key	<input checked="" type="checkbox"/>	

The queried data can be found as a value in the signal *Response body*. You can use a text splitter module to split up the received data and display it in a clear format. See also application example, chapter ↗ *Application example HTTP(S)*, page 24.

3.2.6 Analog and digital signals

The HTTP(S) module provides auto-generated signals.

Analog tab

By default, each HTTP(S) module contains these analog signals in the *Analog* tab:

- Response status code
- Response duration
- Response error type
- Response error message
- Response body
- Endpoint

The signals always contain the values of the last request.

HTTP(S) (3)					
RE ST	General	RE ST	Request settings	Analog	Digital
	Name	Unit	Active	Actual	
0	Response status code		<input checked="" type="checkbox"/>		
1	Response duration		<input checked="" type="checkbox"/>		
2	Response error type		<input checked="" type="checkbox"/>		
3	Response error message		<input checked="" type="checkbox"/>		
4	Response body		<input checked="" type="checkbox"/>		
5	Endpoint		<input checked="" type="checkbox"/>		

Digital tab

Each HTTP(S) module also provides the digital signal *Pending request*. This signal indicates whether a request is currently active (1) or not (0).

HTTP(S) (3)					
RE ST	General	RE ST	Request settings	Analog	Digital
	Name	Unit	Active	Actual	
0	Pending request		<input checked="" type="checkbox"/>		

4 Application example HTTP(S)

Task:

The energy prices for one day are to be retrieved in hourly values from an energy supplier's price portal.

4.1 Preparation

The required parameters and approved methods for the data request were obtained from the portal operator beforehand. An HTTP(S) request was formulated in java script accordingly. The request returns the desired data.

The task is now to configure an HTTP(S) module in *ibaPDA* so that the request works with it.

Example request in java script

```
const url = https://seffaflik.epias.com.tr/electricity-service/v1/markets/dam/data/mcp;  
  
const payload = {  
  startDate: "2024-04-30T00:00:00+03:00",  
  endDate: "2024-04-30T00:00:00+03:00",  
  page: {  
    number: 1,  
    size: 24,  
    sort: {  
      direction: "ASC",  
      field: "date"  
    }  
  }  
};  
  
fetch(url, {  
  method: 'POST',  
  headers: {  
    'Content-Type': 'application/json'  
  },  
  body: JSON.stringify(payload)  
})  
  .then(response => response.json())  
  .then(data => console.log(data))  
  .catch(error => console.error('Error:', error));
```


4.2 Configuration of the ibaPDA HTTP(S) module

You configure the request in the *Request* settings tab.

The following figure shows an example of the java script request and the corresponding settings in the HTTP(S) module.

```

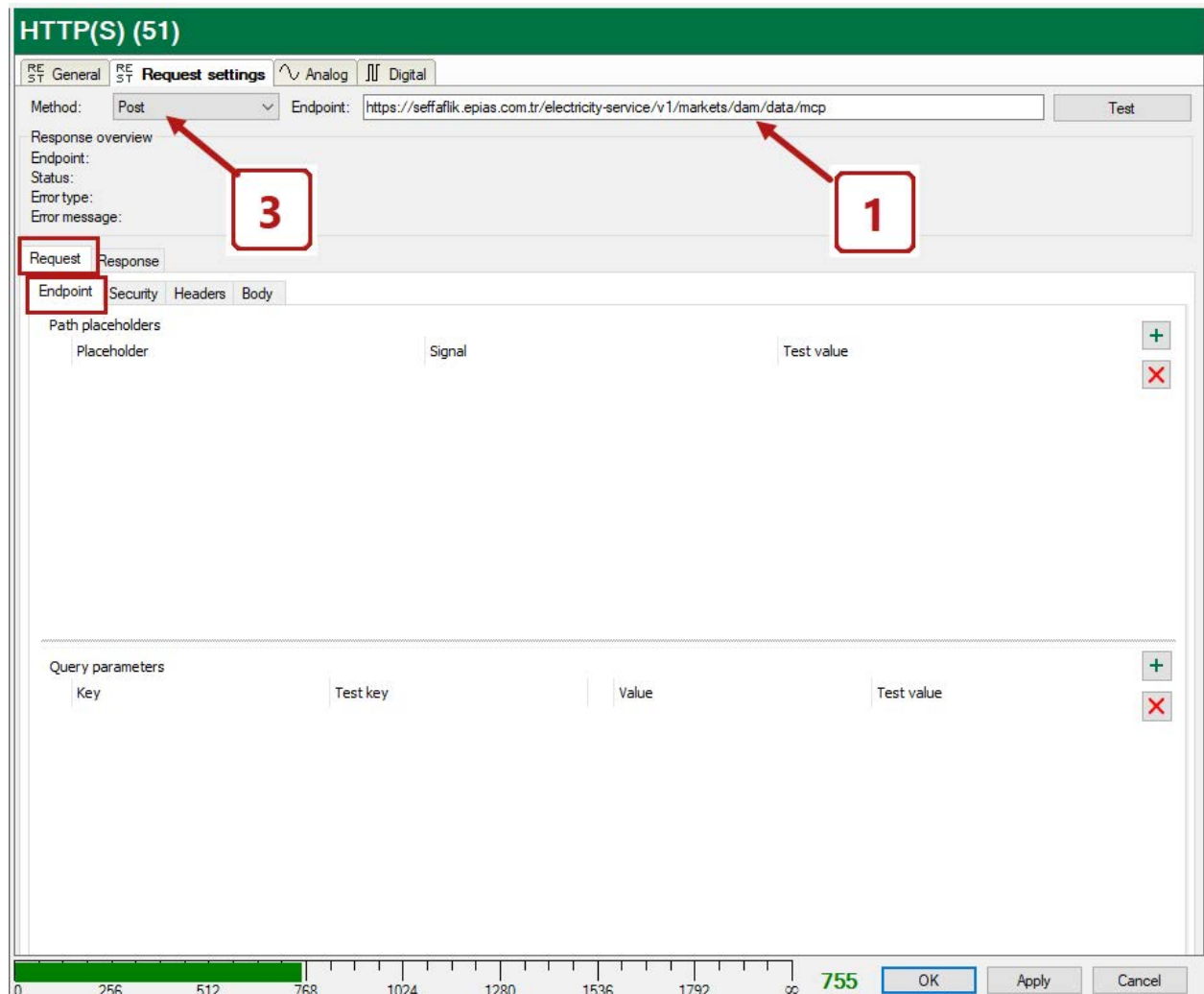
1 const url = https://seffaflik.epias.com.tr/electricity-service/v1/markets/dam/data/mcp;
2 const payload = {
3   startDate: "2024-04-30T00:00:00+03:00",
4   endDate: "2024-04-30T00:00:00+03:00",
5   page: {
6     number: 1,
7     size: 24,
8     sort: {
9       direction: "ASC",
10      field: "date"
11    }
12  }
13 };
14
15 fetch(url, {
16   method: 'POST',
17   headers: {
18     'Content-Type': 'application/json'
19   },
20   body: JSON.stringify(payload)
21 })
22 .then(response => response.json())
23 .then(data => console.log(data))
24 .catch(error => console.error('Error:', error));

```

1	Endpoint
2	Request body (when used in the HTTP(S) module, write in json format, see below)
3	Method
4	Request header
5	Response output (response)

Accordingly, the entries in the module are as follows:

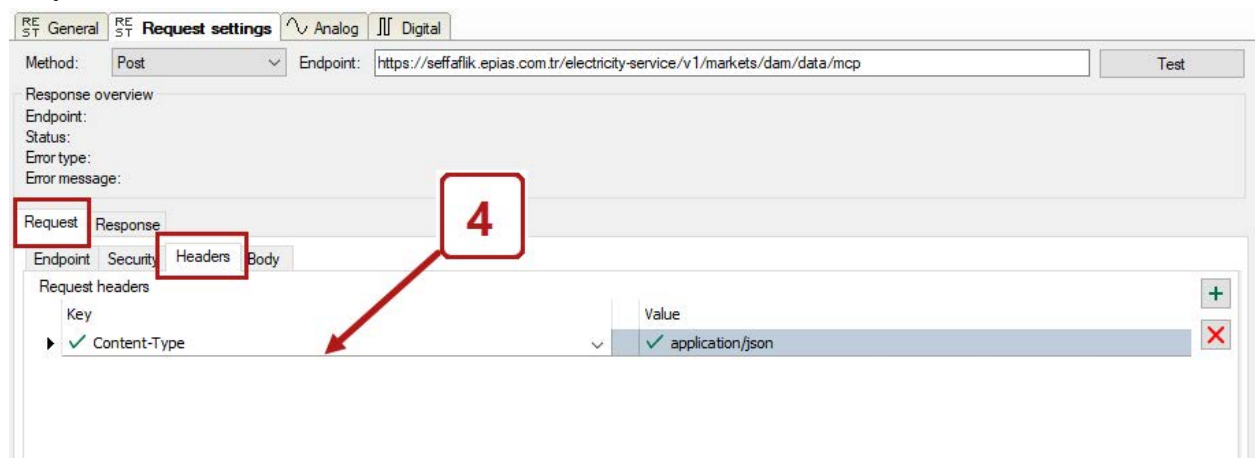
Request settings tab



Path placeholders are not required here, as the absolute path is already entered in full in the endpoint entry.

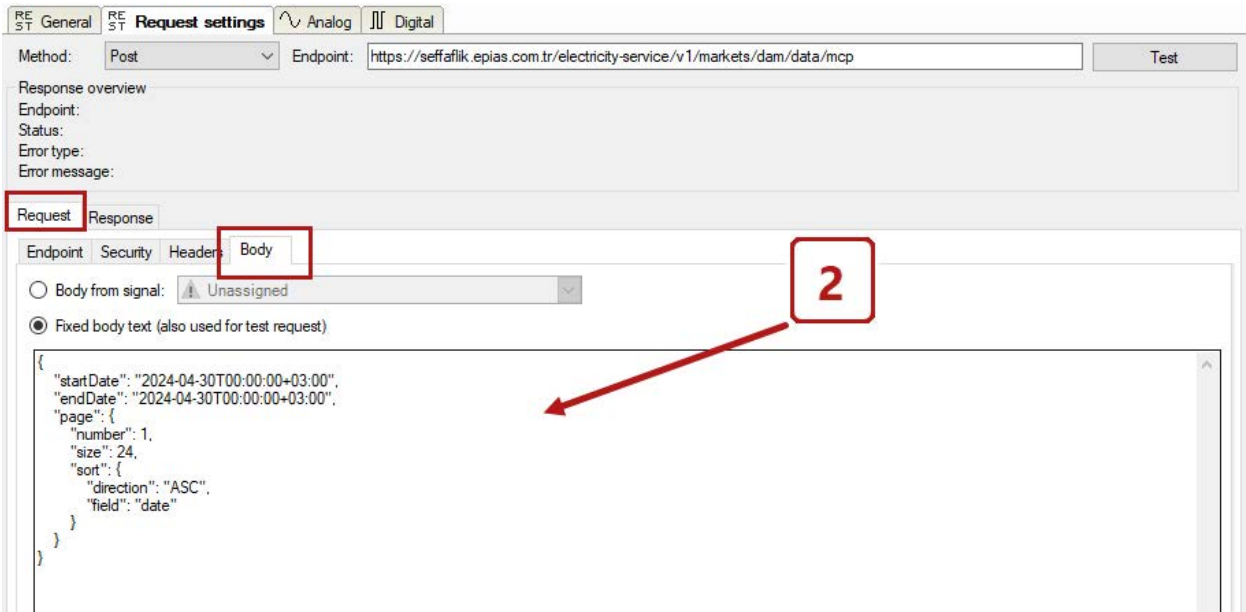
Query parameters are not necessary here. The request is fully specified in the *Body* tab.

Request – Header tab



The key-value pair "Content-Type" specifies here that the request is formulated as a json script.

Request – Body tab



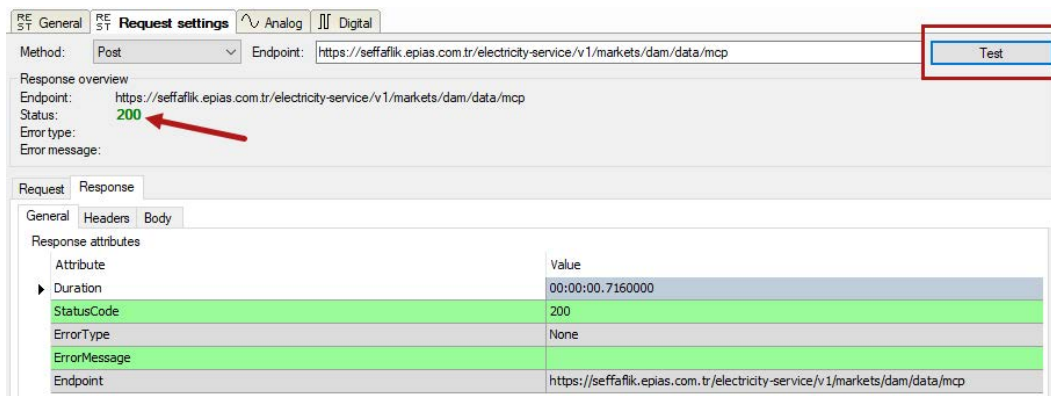
In contrast to the java script version of the request from the example above, the body for the HTTP(S) module must already be formulated in json format.

The request specifically requests the data for 24 hours on 2024-04-30 in ascending order (date/ time).

Test the request

If the entries are complete, a test request can be started by clicking the <Test> button.

If the request was successful, the value **200** appears for the *Status*.



If corresponding data is available on the target system, it is displayed in the *Response - Body* tab.

Request settings window showing the response body content. The response is a JSON array of 24 data sets, each containing a date, hour, price in USD, and price in EUR.

```

{
  "items": [
    {
      "date": "2024-04-30T00:00:00\u002B03:00",
      "hour": "00:00",
      "price": 2399.9899999999998,
      "priceUsd": 73.95,
      "priceEur": 68.91
    },
    {
      "date": "2024-04-30T01:00:00\u002B03:00",
      "hour": "01:00",
      "price": 2363.1399999999999,
      "priceUsd": 72.82,
      "priceEur": 67.86
    },
    {
      "date": "2024-04-30T02:00:00\u002B03:00",
      "hour": "02:00",
      "price": 1631.98,
      "priceUsd": 50.29,
      "priceEur": 46.86
    },
    {
      "date": "2024-04-30T03:00:00\u002B03:00",
      "hour": "03:00",
      "price": 1631.98,
      "priceUsd": 50.29,
      "priceEur": 46.86
    },
    {
      "date": "2024-04-30T04:00:00\u002B03:00",
      "hour": "04:00",
      "price": 1631.98,
      "priceUsd": 50.29,
      "priceEur": 46.86
    }
  ]
}

```

The response provides 24 data sets with the complete date, the full hour of the day and the price information in local currency, USD and EUR.

Note



This example is very simple because it only requests the data for a fixed day. A variable request in which the date can be specified dynamically using signals would be more realistic. For example, you could use a text creator module to dynamically design the request body using text signals and then select the *Body from signal* option in the *Request - Body* tab. The signal would then be a text signal containing the complete body.

4.3 Further processing of the requested data

The most important information is available in the *Analog* tab of the module.

Name	Unit	Active	Actual
0 Response status code		<input checked="" type="checkbox"/>	200
1 Response duration		<input checked="" type="checkbox"/>	00:00:00.1770000
2 Response error type		<input checked="" type="checkbox"/>	None
3 Response error message		<input checked="" type="checkbox"/>	
4 Response body		<input checked="" type="checkbox"/>	{\"items\": [{\"date\": \"2024-04-30T00:00:00+03:00\", \"hour\": \"00:00\", \"price\": 2399.98999999...
5 Endpoint		<input checked="" type="checkbox"/>	https://seffafik.epias.com.tr/electricity-service/v1/markets/dam/data/mcp

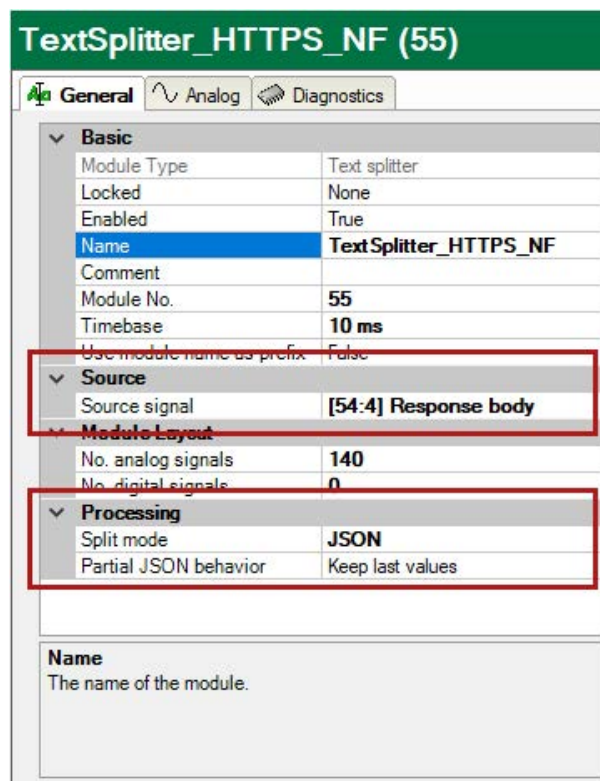
The requested data can be found as a value in the signal *Response body*, in this case a text signal.

To be able to process the individual values as texts or numbers, create a text splitter module in the *Analytics* tab of the I/O Manager. You can use the text splitter module to split the large data block as you need it.

4.4 Configuring the text splitter module

Add a text splitter module and select the text signal with the request result from the HTTP(S) module ("Response Body") as the source signal in the *General* module settings.

As the response was delivered in json format, set the split mode to JSON.



The new text splitter module still has the empty signal grids in the *Analog* tab. However, if data has already been received, the received text is displayed in the *Analog* tab under the signal grid. Now you only need to click on the <Add all JSON values> button and the individual signals are read from the json script and created as analog signals.

TextSplitter_HTTPS_NF (55)

General Analog Diagnostics

Name	Unit	JSON path	Data Type	Filter	Active	Actual
0 items[0].date		items[0].date	STRING	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2024-04-30T00:00:00+03:00
1 items[0].hour		items[0].hour	STRING	<input type="checkbox"/>	<input checked="" type="checkbox"/>	00:00
2 items[0].price		items[0].price	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2399,99
3 items[0].priceUsd		items[0].priceUsd	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	73,95
4 items[0].priceEur		items[0].priceEur	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	68,91
5 items[1].date		items[1].date	STRING	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2024-04-30T01:00:00+03:00
6 items[1].hour		items[1].hour	STRING	<input type="checkbox"/>	<input checked="" type="checkbox"/>	01:00
7 items[1].price		items[1].price	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2363,14
8 items[1].priceUsd		items[1].priceUsd	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	72,82
9 items[1].priceEur		items[1].priceEur	FLOAT	<input type="checkbox"/>	<input checked="" type="checkbox"/>	67,86
10 items[2].date		items[2].date	STRING	<input type="checkbox"/>	<input checked="" type="checkbox"/>	2024-04-30T02:00:00+03:00
11 items[2].hour		items[2].hour	STRING	<input type="checkbox"/>	<input checked="" type="checkbox"/>	02:00

The yellow part highlights the selected part. To change the selected part you can edit the "JSON path" column in the signal grid. You can also change it by clicking on the desired row in the preview while the "JSON path" column is selected.

Add all JSON values

```

{
  "items": [
    {
      "date": "2024-04-30T00:00:00+03:00",
      "hour": "00:00",
      "price": 2399.99,
      "priceUsd": 73.95,
      "priceEur": 68.91
    },
    {
      "date": "2024-04-30T01:00:00+03:00",
      "hour": "01:00",
      "price": 2363.14,
      "priceUsd": 72.82,
      "priceEur": 67.86
    },
    {
      "date": "2024-04-30T02:00:00+03:00",
      "hour": "02:00"
    }
  ]
}
    
```

The values are now available for further processing, e.g. to display them with *ibaQPanel*:

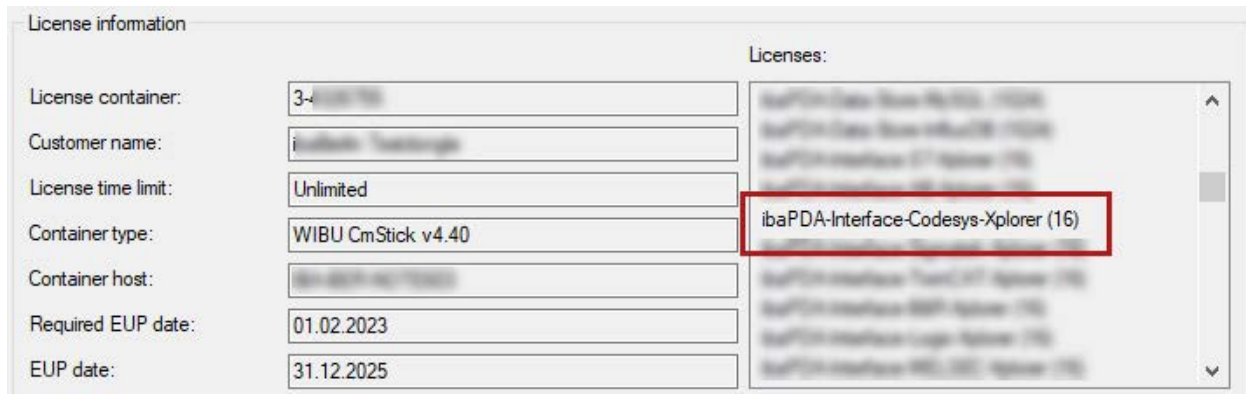
Date	Hour	Price	Price USD	Price EUR
2024-04-30	00:00	2399,99	73,95	68,91
2024-04-30	01:00	2363,14	72,82	67,86
2024-04-30	02:00	1631,98	50,29	46,86
2024-04-30	03:00	1631,98	50,29	46,86
2024-04-30	04:00	1685,00	50,29	46,86
2024-04-30	05:00	1685,00	51,92	48,38
2024-04-30	06:00	999,99	30,81	28,71
2024-04-30	07:00	1631,97	50,29	46,86

5 Diagnostics

5.1 License

If the interface is not displayed in the signal tree, you can either check in *ibaPDA* in the I/O Manager under *General – Settings* or in the *ibaPDA* service status application whether your license for this interface has been properly recognized. The number of licensed connections is shown in brackets.

The figure below shows the license for the *Codesys Xplorer* interface as an example.



5.2 Log files

If connections to target platforms or clients have been established, all connection-specific actions are logged in a text file. You can open this (current) file and, e.g., scan it for indications of possible connection problems.

You can open the log file via the button <Open log file>. The button is available in the I/O Manager:

- for many interfaces in the respective interface overview
- for integrated servers (e.g. OPC UA server) in the *Diagnostics* tab.

In the file system on the hard drive, you can find the log files of the *ibaPDA* server (...\[ProgramData\iba\ibaPDA\Log](#)). The file names of the log files include the name or abbreviation of the interface type.

Files named [interface.txt](#) are always the current log files. Files named [Interface_yyyy_mm_dd_hh_mm_ss.txt](#) are archived log files.

Examples:

- [ethernetipLog.txt](#) (log of EtherNet/IP connections)
- [AbEthLog.txt](#) (log of Allen-Bradley Ethernet connections)
- [OpcUAServerLog.txt](#) (log of OPC UA server connections)

5.3 Error messages

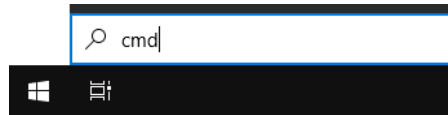
The following warnings and errors may occur during the validation:

Warning/error message	Possible cause or remedy
The endpoint cannot be empty.	The endpoint field must contain a value.
The endpoint must start with http, https or a placeholder	<p>The endpoint field is filled, but doesn't start with http, https or a placeholder.</p> <p>To fix this, prefix the address with http, https or use a placeholder that is filled with either http or https during the acquisition.</p> <p>Examples:</p> <p>http://iba-ag.com</p> <p>https://iba-ag.com</p> <p>{protocol}://iba-ag.com</p>
The placeholder {...} is not assigned to a signal	A placeholder is used but no signal is assigned.

5.4 Connection diagnostics with PING

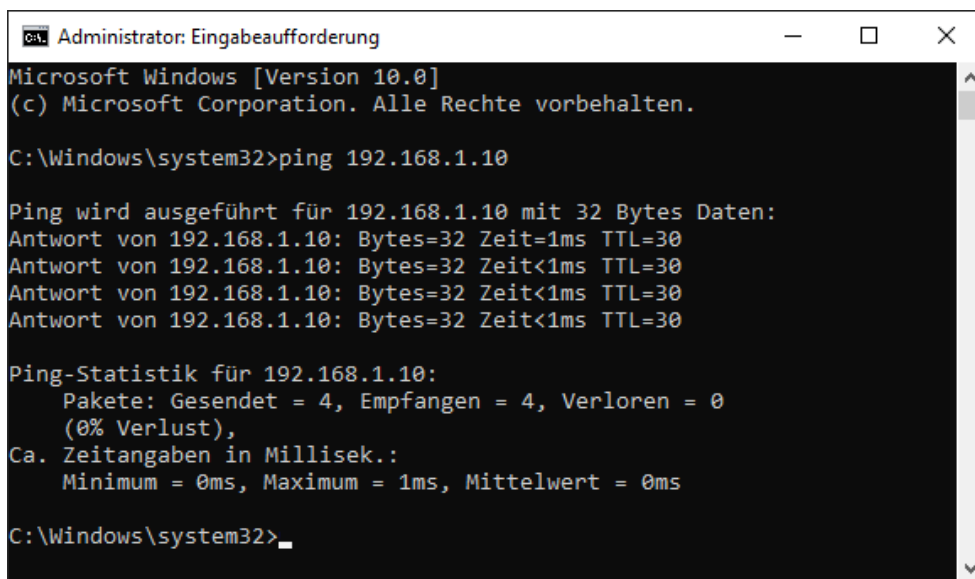
PING is a system command with which you can check if a certain communication partner can be reached in an IP network.

1. Open a Windows command prompt.



2. Enter the command "ping" followed by the IP address of the communication partner and press <ENTER>.

→ With an existing connection you receive several replies.

A screenshot of a Windows command prompt window titled 'Administrator: Eingabeaufforderung'. The window shows the following text:

```
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

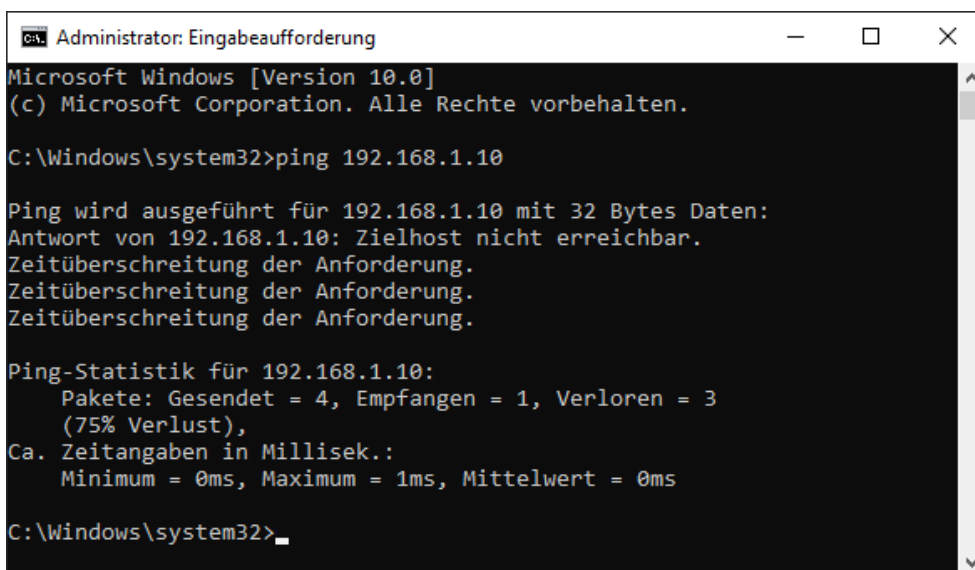
C:\Windows\system32>ping 192.168.1.10

Ping wird ausgeführt für 192.168.1.10 mit 32 Bytes Daten:
Antwort von 192.168.1.10: Bytes=32 Zeit=1ms TTL=30
Antwort von 192.168.1.10: Bytes=32 Zeit<1ms TTL=30
Antwort von 192.168.1.10: Bytes=32 Zeit<1ms TTL=30
Antwort von 192.168.1.10: Bytes=32 Zeit<1ms TTL=30

Ping-Statistik für 192.168.1.10:
    Pakete: Gesendet = 4, Empfangen = 4, Verloren = 0
    (0% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 0ms, Maximum = 1ms, Mittelwert = 0ms

C:\Windows\system32>
```

→ With no existing connection you receive error messages.

A screenshot of a Windows command prompt window titled 'Administrator: Eingabeaufforderung'. The window shows the following text:

```
Microsoft Windows [Version 10.0]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

C:\Windows\system32>ping 192.168.1.10

Ping wird ausgeführt für 192.168.1.10 mit 32 Bytes Daten:
Antwort von 192.168.1.10: Zielhost nicht erreichbar.
Zeitüberschreitung der Anforderung.
Zeitüberschreitung der Anforderung.
Zeitüberschreitung der Anforderung.

Ping-Statistik für 192.168.1.10:
    Pakete: Gesendet = 4, Empfangen = 1, Verloren = 3
    (75% Verlust),
Ca. Zeitangaben in Millisek.:
    Minimum = 0ms, Maximum = 1ms, Mittelwert = 0ms

C:\Windows\system32>
```

5.5 Diagnostic modules

Diagnostic modules are available for most Ethernet based interfaces and Xplorer interfaces. Using a diagnostic module, information from the diagnostic displays (e.g. diagnostic tabs and connection tables of an interface) can be acquired as signals.

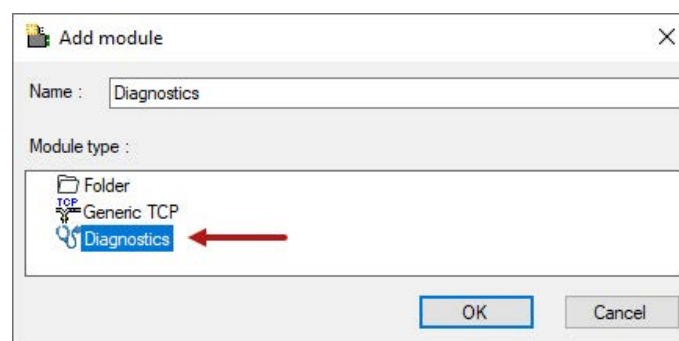
A diagnostic module is always assigned to a data acquisition module of the same interface and supplies its connection information. By using a diagnostic module, you can record and analyze the diagnostic information continuously in the *ibaPDA* system.

Diagnostic modules do not consume any license connections because they do not establish their own connection but refer to another module.

Example for the use of diagnostic modules:

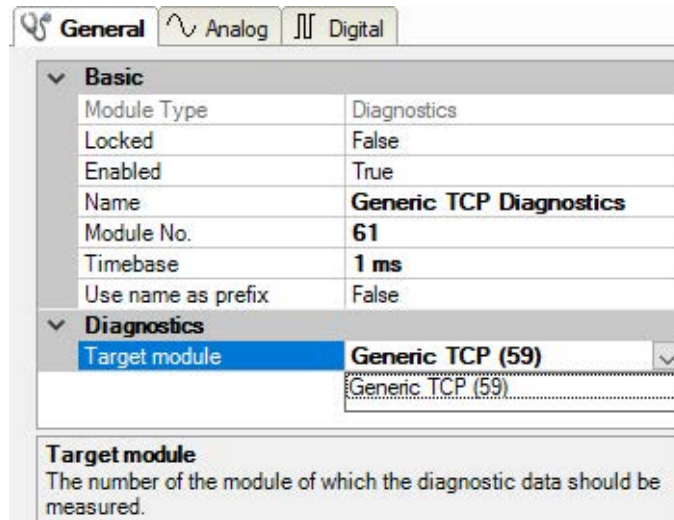
- A notification can be generated, whenever the error counter of a communication connection exceeds a certain value or the connection gets lost.
- In case of a disturbance, the current response times in the telegram traffic may be documented in an incident report.
- The connection status can be visualized in *ibaQPanel*.
- You can forward diagnostic information via the SNMP server integrated in *ibaPDA* or via OPC DA/UA server to superordinate monitoring systems like network management tools.

In case the diagnostic module is available for an interface, a "Diagnostics" module type is shown in the "Add module" dialog (example: Generic TCP).



Module settings diagnostic module

For a diagnostic module, you can make the following settings (example: Generic TCP):



The basic settings of a diagnostic module equal those of other modules.

There is only one setting which is specific for the diagnostic module: the target module.

By selecting the target module, you assign the diagnostic module to the module on which you want to acquire information about the connection. You can select the supported modules of this interface in the drop-down list of the setting. You can assign exactly one data acquisition module to each diagnostic module. When having selected a module, the available diagnostic signals are immediately added to the *Analog* and *Digital* tabs. It depends on the type of interface, which signals exactly are added. The following example lists the analog values of a diagnostic module for a Generic TCP module.

General Analog Digital						
Name	Unit	Gain	Offset	Active	Actual	
0 IP address (part 1)			1	0	<input checked="" type="checkbox"/>	
1 IP address (part 2)			1	0	<input checked="" type="checkbox"/>	
2 IP address (part 3)			1	0	<input checked="" type="checkbox"/>	
3 IP address (part 4)			1	0	<input checked="" type="checkbox"/>	
4 Port			1	0	<input checked="" type="checkbox"/>	
5 Message counter			1	0	<input checked="" type="checkbox"/>	
6 Incomplete errors			1	0	<input checked="" type="checkbox"/>	
7 Packet size (actual)	bytes		1	0	<input checked="" type="checkbox"/>	
8 Packet size (max)	bytes		1	0	<input checked="" type="checkbox"/>	
9 Time between data (actual)	ms		1	0	<input checked="" type="checkbox"/>	
10 Time between data (min)	ms		1	0	<input checked="" type="checkbox"/>	

For example, the IP (v4) address of a Generic TCP module (see fig. above) will always be split into 4 parts derived from the dot-decimal notation, for better reading. Also other values are being determined, as there are port number, counters for telegrams and errors, data sizes and telegram cycle times. The following example lists the digital values of a diagnostic module for a Generic TCP module.

General Analog Digital			
Name	Active	Actual	
0 Active connection mode	<input checked="" type="checkbox"/>		
1 Invalid packet	<input checked="" type="checkbox"/>		
2 Connecting	<input checked="" type="checkbox"/>		
3 Connected	<input checked="" type="checkbox"/>		

Diagnostic signals

Depending on the interface type, the following signals are available:

Signal name	Description
Active	Only relevant for redundant connections. Active means that the connection is used to measure data, i.e. for redundant standby connections the value is 0. For normal/non-redundant connections, the value is always 1.
Buffer file size (actual/avg/max)	Size of the file for buffering statements
Buffer memory size (actual/avg/max)	Size of the memory used by buffered statements
Buffered statements	Number of unprocessed statements in the buffer
Buffered statements lost	Number of buffered but unprocessed and lost statements
Connected	Connection is established
Connected (in)	A valid data connection for the reception (in) is available
Connected (out)	A valid data connection for sending (out) is available
Connecting	Connection being established
Connection attempts (in)	Number of attempts to establish the receive connection (in)
Connection attempts (out)	Number of attempts to establish the send connection (out)
Connection ID O->T	ID of the connection for output data (from the target system to <i>ibaPDA</i>). Corresponds to the assembly instance number
Connection ID T->O	ID of the connection for input data (from <i>ibaPDA</i> to target system). Corresponds to the assembly instance number
Connection phase (in)	Status of the ibaNet-E data connection for reception (in)
Connection phase (out)	Status of the ibaNet-E data connection for sending (out)
Connections established (in)	Number of currently valid data connections for reception (in)
Connections established (out)	Number of currently valid data connections for sending (out)
Data length	Length of the data message in bytes
Data length O->T	Size of the output message in byte
Data length T->O	Size of the input message in byte
Destination IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to <i>ibaPDA</i>)
Destination IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from <i>ibaPDA</i> to target system)
Disconnects (in)	Number of currently interrupted data connections for reception (in)
Disconnects (out)	Number of currently interrupted data connections for sending (out)
Error counter	Communication error counter
Exchange ID	ID of the data exchange
Incomplete errors	Number of incomplete messages

Signal name	Description
Incorrect message type	Number of received messages with wrong message type
Input data length	Length of data messages with input signals in bytes (<i>ibaPDA</i> receives)
Invalid packet	Invalid data packet detected
IP address (part 1-4)	4 octets of the IP address of the target system
Keepalive counter	Number of KeepAlive messages received by the OPC UA Server
Lost images	Number of lost images (in) that were not received even after a retransmission
Lost Profiles	Number of incomplete/incorrect profiles
Message counter	Number of messages received
Messages per cycle	Number of messages in the cycle of the update time
Messages received since configuration	Number of received data telegrams (in) since start of acquisition
Messages received since connection start	Number of received data telegrams (in) since the start of the last connection setup. Reset with each connection loss.
Messages sent since configuration	Number of sent data telegrams (out) since start of acquisition
Messages sent since connection start	Number of sent data telegrams (out) since the start of the last connection setup. Reset with each connection loss.
Multicast join error	Number of multicast login errors
Number of request commands	Counter for request messages from <i>ibaPDA</i> to the PLC/CPU
Output data length	Length of the data messages with output signals in bytes (<i>ibaPDA</i> sends)
Packet size (actual)	Size of the currently received message
Packet size (max)	Size of the largest received message
Ping time (actual)	Response time for a ping telegram
Port	Port number for communication
Producer ID (part 1-4)	Producer ID as 4-byte unsigned integer
Profile Count	Number of completely recorded profiles
Read counter	Number of read accesses/data requests
Receive counter	Number of messages received
Response time (actual/average/max/min)	Response time is the time between measured value request from <i>ibaPDA</i> and response from the PLC or reception of the data. Actual: current value Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Retransmission requests	Number of data messages requested again if lost or delayed

Signal name	Description
Rows (last)	Number of resulting rows by the last SQL query (within the configured range of result rows)
Rows (maximum)	Maximum number of resulting rows by any SQL query since the last start of acquisition (possible maximum equals the configured number of result rows)
Send counter	Number of send messages
Sequence errors	Number of sequence errors
Source IP address (part 1-4) O->T	4 octets of the IP address of the target system Output data (from target system to <i>ibaPDA</i>)
Source IP address (part 1-4) T->O	4 octets of the IP address of the target system Input data (from <i>ibaPDA</i> to target system)
Statements processed	Number of executed statements since last start of acquisition
Synchronization	Device is synchronized for isochronous acquisition
Time between data (actual/ max/min)	Time between two correctly received messages Actual: between the last two messages Max/min: statistical values since start of acquisition or reset of counters
Time offset (actual)	Measured time difference of synchronicity between <i>ibaPDA</i> and the <i>ibaNet-E</i> device
Topics Defined	Number of defined topics
Topics Updated	Number of updated topics
Unknown sensor	Number of unknown sensors
Update time (actual/average/ configured/max/min)	Specifies the update time in which the data is to be retrieved from the PLC, the CPU or from the server (configured). Default is equal to the parameter "Timebase". During the measurement the real actual update time (actual) can be higher than the set value, if the PLC needs more time to transfer the data. How fast the data is really updated, you can check in the connection table. The minimum achievable update time is influenced by the number of signals. The more signals are acquired, the greater the update time becomes. Average/max/min: static values of the update time since the last start of the acquisition or reset of the counters.
Write counter	Number of successful write accesses
Write lost counter	Number of failed write accesses

6 Support and contact

Support

Phone: +49 911 97282-14

Email: support@iba-ag.com

Note



If you need support for software products, please state the number of the license container. For hardware products, please have the serial number of the device ready.

Contact

Headquarters

iba AG
Koenigswarterstrasse 44
90762 Fuerth
Germany

Phone: +49 911 97282-0

Email: iba@iba-ag.com

Mailing address

iba AG
Postbox 1828
D-90708 Fuerth, Germany

Delivery address

iba AG
Gebhardtstrasse 10
90762 Fuerth, Germany

Regional and Worldwide

For contact data of your regional iba office or representative please refer to our web site:

www.iba-ag.com