CMe3100 Plugin JSON-RPC API

CONTENTS

1 Document notes

All information in this manual, including product data, diagrams, charts, etc. represents information on products at the time of publication, and is subject to change without prior notice due to product improvements or other reasons. It is therefore recommended that customers contact Elvaco AB for the latest product information before purchasing a CMe Series product.

The documentation and product are provided on an “as is” basis only and may contain deficiencies or inadequacies. Elvaco AB takes no responsibility for damages, liabilities or other losses by using this product.

1.1 Copyright and Trademark

1.2 Contacts

Elvaco AB Headquarter

Teknikgatan 18
434 37 Kungsbacka
SWEDEN

Phone: +46 300 30250
Fax: +46 300 18440

E-Mail: info@elvaco.com

Elvaco AB Technical Support

Phone: +46 300 434300
E-Mail: support@elvaco.se

Online: http://www.elvaco.com

2 Using this manual

2.1 Purpose and Audience

Elvaco JSON-RPC API is for developers and administrators who needs to read actual data from the CMe3100. A normal application would be a product that needs to read temperature sensor values, and later use this information in energy calculation etc.

2.2 Models

CMe3100

2.3 Additional and updated information

Latest documentation version is available on Elvaco web site at http://www.elvaco.com.

3 Introduction

This document describes the Elvaco JSON-RPC API and interactions within resources provided by the CMe3100 product. Elvaco JSON-RPC API provides access to measurement series and the actual data in the measurement series. All actions through API are done by using standard HTTP method POST. Standard HTTP response codes used to indicate success and error conditions. The JSON-RPC API only provides read access to the measurement series and the current measurement serie data.

3.1 Preconditions

The CMe3100 needs to have a license for Elvaco JSON-RPC plugin. Information about the installed license can be found in the section /About/Licenses in the CMe3100 web interface.

The Elvaco-JSON-RPC plugin also needs to be installed. The plugin is by default disabled. The plugin can be enabled in the section /Settings/Add-ons in the CMe3100 web interface.

If the product has no license for Elvaco JSON-RPC plugin, please contact Elvaco AB or your local partner to buy a license.

The Elvaco JSON-RPC API can be accessed on the following URL:<CMe3100 URL>/Elvaco-JSON-RPC/PDB.

3.2 Configuration of the Elvaco JSON-RPC API

The only configuration that can be done is regarding the HTTP authentication method. This setting can be changed in the Settings of the Elvaco JSON-RPC plugin in the web interface.

Available settings are *none* or *basic*. When *none* is selected, no security is enabled in the Elvaco JSON-RPC API, and all methods can be used by anyone who has access to the product. When *basic* is selected, the Elvaco JSON-RPC API is protected by HTTP basic authentication and the username and password must be provided to access the API. Users can be added in the /Settings/Users section in the CMe3100 web interface.

3.3 Syntax

The Elvaco JSON-RPC API is based on HTTP POST methods with actual JSON-RPC method and parameters in the BODY of the request. The request from the client to the server (CMe3100) must include the header "*Content-Type: application/json-rpc".* The body contents are formatted in JSON objects.

3.3.1 The request object

The request object contains the following attributes, which describes the method to execute.

Attribute	Value(s)	Description	Mandatory
"jsonrpc"	"2.0"	Static value. Should be set to "2.0".	Yes
"method"	"pdb.browse" "pdb.getvalue"	"pdb.browse" will return a list of available data points. "pdb.getvalue" is used to get the actual value of one or more data points.	Yes
"params"	Variable list of parameters depending on "method"	Used as a parameter list to the method.	No

“id”	User definable identity for a specific request.	The “id” and the user defined value will be returned in the response object.	Yes

Table 1 Request object attribute

3.3.2 The response object

The response object contains the following attributes, which describes the return value of the executed method.

Attribute	Value(s)	Description	Mandatory
“jsonrpc”	“2.0”	Static value. Should be set to “2.0”.	Yes
“result”	Object with result attributes.	Depends on the executed method.	Yes
“id”	User definable identity for a specific request.	The “id” is the user defined value which was set .	Yes

Table 2 Response object attribute

3.4 Methods

This section describes the available methods in Elvaco JSON-RPC API.

3.4.1 pdb.browse

The pdb.browse is used to retrieve a list of available data points (measurement series). The available data points can later be read by using the method pdb.getvalue.

The parameter “params” are for future used to filter specific data points.

Origin	Body
Client	{ "jsonrpc": "2.0", "method": "pdb.browse", *"params": {"limit":200, "offset":0, "language":"sv/en", "filter":"WHERE pid LIKE '%temp%'"},* "id": **ID** } OR { "jsonrpc": "2.0", "method": "pdb.browse", *[200, 0, "sv/en", "WHERE pid LIKE '%temp%'"],* "id": **ID** }
Server	{ "jsonrpc": "2.0", "result":

	{ "devid":"**DEVICEID**", "points":[{ "pid": "**APIIDENTIFIER**", "desc":"**DESCRIPTION**", "acc":"RO", "type":"number", "attr": "**UNIT**" }] }, "id": **ID** } Where: **DEVICEID:** Hostname of the CMe3100 **APIDENTIFIER**: The API Identifier of a measurement serie **DESCRIPTION**: The Description field of the measurement serie **UNIT**: The Unit of the measurement serie **ID**: Used definable identification for the request

Table 3 Method pdb.broswe syntax

An example of a client request and a server response:

Origin	Body
Client	{ "jsonrpc": "2.0", "method": "pdb.browse", "id": 1 }
Server	{ "jsonrpc": "2.0", "result": { "devid":"CMe3100-0016000001", "points":[{ "pid": "**met1**", "desc":"**Temperature value**", "acc":"RO", "type":"number", "attr": "**°C**" }] }, "id": 3 }

Table 4 pdb.browse example

3.4.2 pdb.getvalue

The pdb.getvalue is used to read the actual value for one or more data points (measurement series).

Origin	Body

Client	{ "jsonrpc": "2.0", "method": "pdb.getvalue", "params": ["**POINT1**","**POINT2**","**POINTn**"] "id": **ID** } OR { "jsonrpc": "2.0", "method": "pdb.getvalue", "params": [["**POINT1**","**POINT2**","**POINTn**"]] "id": **ID** }
Server	{ "jsonrpc": "2.0", "result": { "timet":"**TICK**", "times": "**DATETIME**" "points": [{ "pid":"**POINT1**","value":**VALUEP1** }, { "pid":"**POINT2**","value": **VALUEP2** }, { "pid":"**POINTn**","value": **VALUEn** },] "error":"null", }, "id": **ID** } Where: **POINT1..N**: Name of data point (measurement serie api identification) **VALUE1..N**: The value of the data point (measurement serie api identification) **TICK:** Number of milliseconds since 1970-01-01 00:00 **DATETIME:** Date/time in string format; YYYY-MM-DD hh:mm:ss **ID**: Used definable identification for this request

Table 5 Method pdb.getvalue syntax

An example of a client request and a server response:

Origin	**Body**
Client	{ "jsonrpc": "2.0", "method": "pdb.getvalue",

	"params": ["**met1**","**met2**","**met3**"] "id": **1** } OR { "jsonrpc": "2.0", "method": "pdb.getvalue", "params": [["**met1**","**met2**","**met3**"]] "id": **1** }
Server	{ "jsonrpc": "2.0", "result": { "timet":"**1398241800001**", "times": "**2014-04-23 10:30:00**" "points": [{ "pid":"**met1**","value":**12** }, { "pid":"**met2**","value": **14** }, { "pid":"**met3**","value": **1** },] "error":"null", }, "id": **1** }

Table 6 pdb.getvalue example

3.4.3 pdb.version

The pdb.version is used to get the Elvaco JSON-RPC version information.

Origin	Body
Client	{ "jsonrpc": "2.0", "method": "pdb.version", "id": **ID** }
Server	{ "jsonrpc": "2.0", "result": "**VERSION**" "id": **ID** } Where: **ID**: Used definable identification for this request

	VERSION: Version string in the format Major.Minor.

Table 7 Method pdb.version syntax

An example of a client request and a server response:

Origin	Body
Client	{ "jsonrpc": "2.0", "method": "pdb.version", "id": **1** }
Server	{ "jsonrpc": "2.0", "result": "**1.1**" "id": **1** }

Table 8 pdb.version example

Document History

Version	Date	Description	Author
1	2015-06-04	First draft	David Vonasek

3.5 Document software appliance

Type	Version	Date	Comments
CMe3100	>1.2.0	2015-04-07	

4 References

4.1 References

[1]

[2]

4.2 Terms and Abbreviations

Abbreviation	Description

4.2.1 Number representation

Decimal numbers are represented as normal number, i.e. 10 (ten).

Hexadecimal numbers are represented with prefix 0x, i.e. 0x0A (ten)

Binary numbers are represented with prefix 0b, i.e. 0b00001010 (ten)