

NetMeter-3P-600

3-Phase Commercial/Industrial Energy Meter/Monitor with Integrated Networking

NETWORK API GUIDE

Copyright © 2011 Z3 Controls Inc.

CONTENTS

1	Important Notice.....	3
2	Purpose.....	3
3	Introduction	3
3.1	License	3
3.2	Overview	4
4	Quick Start Tutorial.....	4
4.1	Connect to the NetMeter-3P-600.....	4
4.2	Query Real-time NetMeter Data	4
4.3	Access NetMeter Data from a Program	6
5	API Command Format.....	7
5.1	Data Types.....	8
5.1.1	Writing System Parameters	8
5.2	Sensor Time	9
5.3	Command Limitations.....	10
5.4	Return Data	10
5.5	Modeless Operation	11
6	API Command Reference.....	12
6.1	Gateway Specific Commands.....	12
6.1.1	Gateway Network Configuration: gnconfig.json	12
6.2	Energy Sensor Specific Commands	16
6.2.1	Sensor Information: sinfo.json.....	16
6.2.2	Sensor Data Query: sdata.json.....	20
6.2.2.1	Sensor Data Query: Realtime Data (Mode 1)	23
6.2.2.2	Sensor Data Query: Realtime Data (Mode 2)	25
6.2.2.3	Sensor Data Query: Realtime Data (Mode 3)	26
6.2.2.4	Sensor Data Query: Recent History (Modes "f0", "f0a", "f0b").....	27
6.2.2.5	Power Use History: 1 Minute Resolution (Mode "f1t")	29
6.2.2.6	Accumulated Energy History: 1 Hour Resolution (Mode "f1h")	33
6.2.2.7	Dashboard Statistics: (Mode "stat").....	35
6.2.3	Datalog Data Query: datalog.json	37
6.2.3.1	Complete Datalog Data Query.....	39
6.2.3.2	Incremental Datalog Data Query.....	42

1 Important Notice

During normal use, potentially lethal voltages are connected to the NetMeter hardware. Consequently, the NetMeter hardware module should only be installed and serviced by a qualified electrician.

Please read and follow the Installation Manual for all guidelines and safety procedures associated with the installation and standard operation of this hardware. Any specific application of the NetMeter system should be in accordance with your local standards and practices.

Under no circumstances will Z3 Controls Inc. (Z3 Controls) be responsible or liable for any direct, indirect, or circumstantial damages associated with the usage or application of this equipment. No patent liability will be assumed or associated with Z3 Controls with respect to the usage of information, equipment, circuitry, software or practices described within this manual.

2 Purpose

The purpose of this document is to describe the parameters and operation of the Application Programming Interface (the 'API') which allows client applications to access NetMeter-3P data.

Since the Z3 Controls NetMeter already provides the user with a fully-integrated, full-featured graphical user interface ('GUI'), most setups will not require the API. However, the API is invaluable in situations where the NetMeter is intended to be used with other systems.

Potential applications of the API may include:

- Integration into factory automation or building automation systems
- The aggregation of NetMeter data with data from other systems
- Special applications that harvest NetMeter data for various purposes
- Generating custom reports or custom dashboard displays

3 Introduction

3.1 License

The API is designed to enable interested parties, such as third-party developers, small businesses, technical enthusiasts and students, to incorporate a Z3 Controls product into their custom application as a potential solution. Although the API is the intellectual property of Z3 Controls, Z3 Controls grants customers the right to use the API free of charge, under the terms of the license agreement found on the Z3 Controls website, www.z3controls.com.

Users are only permitted to use the API commands specified within this documentation. Any additional, or undocumented, API commands may not be incorporated into a user's application(s) without first obtaining the written permission of Z3 Controls.

Please familiarize yourself with the license agreement before using this device.

3.2 Overview

The API is based on HTTP communications where the Z3 Controls device is considered to be an HTTP server. An HTTP client is necessary in order to make use of this device. For example, this may be achieved in the form of JavaScript using a web browser or by using a programming language (such as C, C + +, PHP, Python, etc.) that has socket libraries which enable TCP/IP HTTP client communication.

Additionally, each Z3 Controls NetMeter device can be logically or physically composed of up to 3 parts:

1. **The Gateway:** this is the component of the device containing the HTTP server. The Gateway component has Gateway specific API commands, such as the Gateway Network Setup which can be either Ethernet or WiFi.
2. **The Sensor(s):** each Z3 Controls Gateway will have a sensor, or input/output signal, attached to it –there may be multiple sensors or I/Os on a single Gateway.
3. **Data logging and storage:** Z3 Controls sensors include data storage and logging. Note that the Control modules do not typically contain storage.

The Gateway and Sensor(s) may be composed of a single unit or they may be physically separate units. For the purposes of the API, they are treated as logically separate units.

The NetMeter-3P is a single physical unit that combines gateway, sensor, and storage.

4 Quick Start Tutorial

This section is for those who wish to start programming with the API right away. If anything is unclear in this tutorial, please skip ahead to the subsequent chapters that explain the operation in greater detail.

4.1 Connect to the NetMeter-3P-600

Follow the instructions in the NetMeter-3P-600 *Installation and Instruction Manual* to set up network communication to the device.

By default, the NetMeter will have a NETBIOS name of NETMETER and will be available at:

<http://netmeter/>

From this point on, it is assumed that your NetMeter has the basic configuration setup and is able to display real-time power from the standard web interface using a web browser.

It is also assumed that the default NETBIOS name (“NETMETER”) is being used. If you have changed it, please substitute your new NETBIOS name as required in the process or access it through the direct IP address.

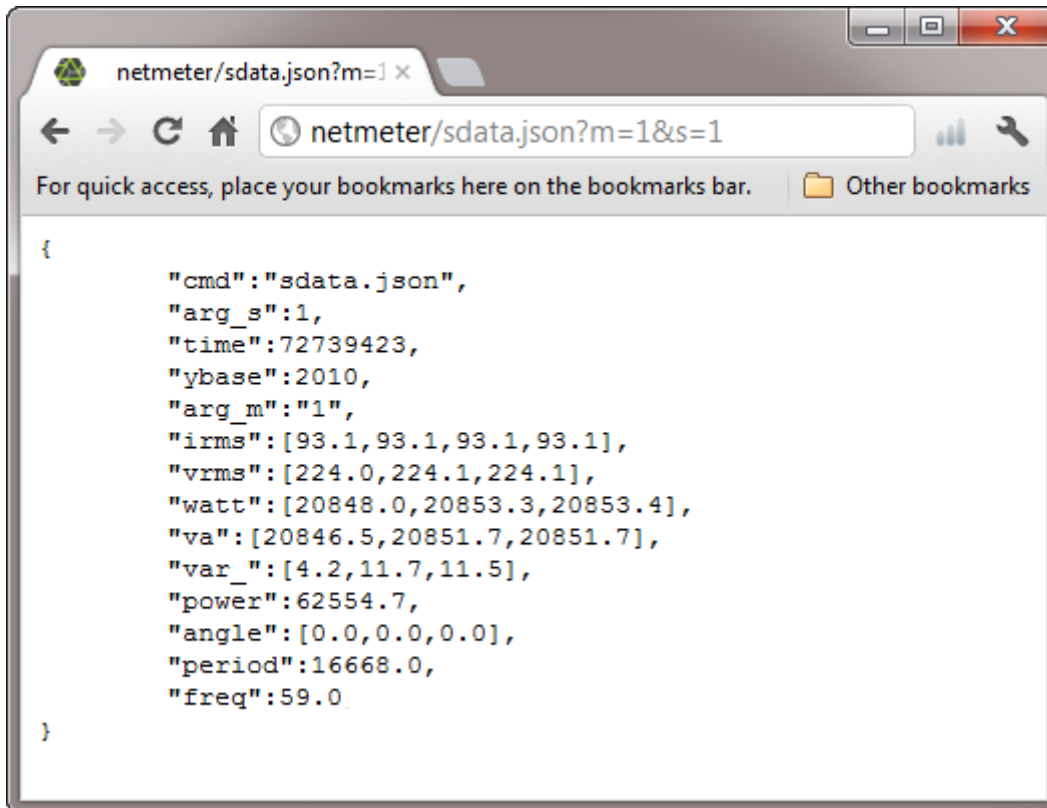
4.2 Query Real-time NetMeter Data

The sdata.json query is used for obtaining many types of data. A “query” is a request for data from the NetMeter and it uses the standard URL format used in web applications.

Enter the following query into the address bar of your web browser:

<http://netmeter/sdata.json?m=1&s=1>

The following data should be displayed (please note that the results may not be exactly as shown):



```
{
  "cmd": "sdata.json",
  "arg_s": 1,
  "time": 72739423,
  "ybase": 2010,
  "arg_m": "1",
  "irms": [93.1, 93.1, 93.1, 93.1],
  "vrms": [224.0, 224.1, 224.1],
  "watt": [20848.0, 20853.3, 20853.4],
  "va": [20846.5, 20851.7, 20851.7],
  "var_": [4.2, 11.7, 11.5],
  "power": 62554.7,
  "angle": [0.0, 0.0, 0.0],
  "period": 16668.0,
  "freq": 59.0
}
```

Figure 1: Example Result from the sdata.json Query

The resulting data structure in Figure 1 above is in JSON format: a lightweight open standard designed for machine and human-readable data interchange. It is language-independent, with parsers available for several languages.

The command has “m = 1” (mode 1) which outputs the instantaneous values for all non-cumulative parameters. They are:

- “irms”: the rms current in Amps for each of the 4 CT inputs (IA,IB,IC,ID)
- “vrms”: the rms voltage in Volts for each of the 3 voltage inputs(VA,VB,VC)
- “watt”: the actual power in Watts for each of the 3 voltage/current inputs
- “va”: the apparent power in Volt-amperes for the 3 voltage/current inputs
- “var_”: the reactive power in Watts for the 3 voltage/current inputs
- “power”: the total power in Watts for all of the active input phases combined (the sum of “watt”)
- “angle”: the phase angle in degrees for the 3 voltage/current inputs
- “period” is the period in microseconds of the AC cycle on VA
- “freq” is the frequency in Hz of the AC cycle on VA (line frequency)

The parameter “s=1” results in all the values above being displayed in standard units (Volts, Amps, Watts, and so on). Using the “s” parameter is the simplest method to get quick results.

4.3 Access NetMeter Data from a Program

Instead of a web browser, your preferred programming language may also be used to execute the above query and then access the returned data structure. Nearly all available computer languages provide for TCP/IP socket support, either natively or through readily available libraries.

Consider the popular server side scripting language PHP: the cURL library can be used for just this purpose. The library is readily available, although you may need to enable this feature in your PHP setup.

Listing 1 is a simple PHP program that performs the query using the cURL library.

Listing 1: PHP Code to Perform Simple NetMeter Query

```
1 <?php
2
3 $Url = 'http://netmeter/sdata.json?m=1&s=1';
4 echo curl_download($Url);
5
6 function curl_download($Url)
7 {
8
9     // is cURL installed yet?
10    if (!function_exists ('curl_init')){
11        die ('Sorry cURL is not installed!');
12    }
13
14    // OK cool - then let's create a new cURL resource handle
15    $ch = curl_init();
16
17    // Now set some options (most are optional)
18
19    // Set URL to download
20    curl_setopt($ch, CURLOPT_URL, $Url);
21
22    // Include header in result? (0 = yes, 1 = no)
23    curl_setopt($ch, CURLOPT_HEADER, 0);
24
25    // Should cURL return or print out the data? (true = return, false = print)
26    curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
27
28    // Timeout in seconds
29    curl_setopt($ch, CURLOPT_TIMEOUT, 10);
30
31    // Download the given URL, and return output
32    $output = curl_exec($ch);
33
34    // Close the cURL resource, and free system resources
35    curl_close($ch);
36
37    return $output;
38 }
39
40 ?>
```

In Listing 1, the main program is only 2 lines of code (Line 3 and 4). The function `curl_download($Url)` is a generic function that can be used to perform a query and return the results as a text string.

Rather than use the data from the query in string format, it can be made available as a data object such that the PHP program can use it more easily. This is illustrated in Listing 2 below. Note that the `curl_download($url)` function has been pushed out to an external include file for convenience.

The PHP function `json_decode()` on line 6 is used to set the variable `$sdata` as a PHP data object.

Listing 2: PHP Code to Perform Simple NetMeter Query and Parse the JSON String

```
1 <?php
2
3 include_once "curl_download.php";
4
5 $url = 'http://netmeter/sdata.json?m=1&s=1';
6 $sdata=json_decode(curl_download($url));
7 echo "Active Power Total    = ". $sdata->power. " Watts\n";
8 echo "Active Power Phase A = ". $sdata->watt[0]. " Watts\n";
9 echo "Active Power Phase B = ". $sdata->watt[1]. " Watts\n";
10 echo "Active Power Phase C = ". $sdata->watt[2]. " Watts\n";
11
12 ?>
```

Running the above PHP script will return results similar to:

```
Active Power Total    = 61594.9 Watts
Active Power Phase A = 20528.8 Watts
Active Power Phase B = 20534.7 Watts
Active Power Phase C = 20531.5 Watts
```

The examples above illustrate how easy it is to incorporate data from the NetMeter into another application with just a few lines of code. The PHP example can easily translate into other programming languages since cURL and JSON libraries exist for most of the popular languages in current use.

5 API Command Format

There are two basic types of API command:

1. **Read Commands:** Commands that query the state of the Gateway or Sensor(s)
2. **Write Commands:** Commands that configure the Gateway or Sensor(s)

Commands to the Gateway are in the form of HTTP: 'GET' or 'POST' requests over the Ethernet/WiFi interface. 'GET' requests take this general form:

```
http://netmeterpath/command?var1=aaa&var2=bbb&var3=ccc
```

Where:

“netmeterpath” is the full path to the NetMeter device or its IP address. The default NETBIOS name for the NetMeter is “netmeter” when using the factory default network setup.

“command” is the API defined command name such as “sinfo.json” for sensor setup information or “sdata.json” for sensor data

“var1”, “var2”, “var3” are parameter names that the command processes

“aaa”, “bbb”, “ccc” are values for the parameters

Note that the order in which parameters are present in the query has no effect on the behaviour of the command.

5.1 Data Types

Parameters are case sensitive. Generally, they are lower case unless otherwise noted. Parameters may be one of those described in Table 1.

Table 1: Mnemonics for Parameter Data Types

Mnemonic	Description
U8	Unsigned value between 0 and 255 (8 bits). Hex values can be used with a "0x" prefix
U16	Unsigned value between 0 and 65,535 (16 bits)
U32	Unsigned value between 0 and 4,294,967,295 (32 bits)
INT8	Signed 8 bit value between -128 to 127
INT16	Signed 16 bit value between -32,768 to 32,767
INT32	Signed 32 bit value between -2147483648 to 2147483647
INT48	Signed 48 bit value between -140737488355328 to 140737488355327
F32	32 bit floating point number
F64	64 bit floating point number
BOOL	Boolean: 0 or 1
STR	Text string: a maximum of 46 characters unless otherwise noted.
MAC	A string in the format of a MAC (Media Access Controller) address such as "00:04:A3:AB:12:34"
IP	A string formatted as an IP address such as "192.168.1.1"
(W)	The (W) annotation indicates that a parameter may also be written. See Section 5.1.1 for additional information.

5.1.1 Writing System Parameters

System parameters are parameters that change the state of the NetMeter and are stored in non-volatile memory. Non-system parameters are only relevant to a specific query.

Most applications designed to access NetMeters directly will not need to write NetMeter system parameters. If required, however, system parameters are written in a similar fashion to that of non-system parameters, using the HTTP URL encoded 'GET' query as demonstrated below:

```
http://netmeterpath/command?parameter=value
```

Where:

“parameter” is the name of the parameter to be written.

“value” is the numeric or string data to be written to the parameter. “value” must be a valid URL/URI encoded value where special characters are escaped as defined by the W3C (World Wide Web Consortium) RFC 3986.

URLs can only be sent over the Internet using the ASCII character-set.

Since URLs often contain characters outside the ASCII set, the URL must be converted into a valid ASCII format.

URL encoding replaces the 'invalid' ASCII characters with a "%" followed by two hexadecimal digits.

URLs cannot contain spaces. URL encoding normally replaces a space with a + sign.

A URL/URI encoding function is provided in libraries for many programming languages. In JavaScript, a valid query may be constructed as:

```
var query="http://netmeter/info.json?" +pname+"="+encodeURIComponent(pvalue);
```

Where:

“pname” is a variable containing the parameter name

“pvalue” is a variable containing the parameter value

“encodeURIComponent()” is a function built into JavaScript that will safely escape “pvalue”

In PHP the same query is constructed as:

```
$query='http://netmeter/info.json?'. $pname.'=' . rawurlencode($pvalue);
```

Parameters may be erased to their default value by setting them to the backspace character (encoded as "%08"). For example:

```
http://netmeterpath/command?parameter=%08
```

Typically, parameters may only be written when the client is authenticated. Authentication may be included in the URL by using the URL encoded method:

```
http://username:password@netmeter/...
```

For security purposes, this method should only be used by scripts over a secure local/VPN network.

5.2 Sensor Time

Some queries input time as a parameter or return time values as data. Time stamps are 32 bit unsigned numbers that work as follows:

- The NetMeter maintains a temperature compensated internal battery backed real time clock that is used to timestamp data.

- Time values used by the NetMeter are the number of seconds since zero hour of January 1 of the “base year” and referenced to the UTC time zone. The base year is defined by the “ybase” value available through the “sinfo.json” (sensor info) query. In the case where ybase = “2010”, a time value of zero would be 00:00:00 (midnight) of January 1, 2010 of the UTC time zone. A time value of 1 is exactly 1 second past that and so on.
- Internally, the NetMeter sensor always operates in UTC time (Zulu Time Zone). For the NetMeter built-in web application, time as shown in the user interface is localized by the application software running in the web browser using whichever localization rules the client device happens to be using. Consequently, a user accessing a NetMeter from a laptop in the EST time zone will see that the current sensor time, as reported by the GUI, is the same as their laptop. Another user with a laptop configured to a different time zone will see that the current time matches their alternate local time. However, the numerical value transmitted by the NetMeter will be exactly the same in both time zones.

Conversion of NetMeter time to the time format of an application language is fairly straightforward. For example, JavaScript time is measured by the number of milliseconds occurred since midnight, January 1, 1970. Consequently, JavaScript time is calculated by multiplying the NetMeter time by 1000 and adding the number of milliseconds from midnight January 1, 1970 until midnight January 1 of ybase. The following JavaScript functions may be used for this purpose:

```
// Convert the Z3 Controls NetMeter time format into a JavaScript style serial number
function fromZ3Time(z3time, ybase)
{
    var UTCOffset = parseInt(Date.UTC(ybase, 0, 1)); // number of ms to add to time stamps
    return z3time*1000+UTCOffset;
}

// Convert a JavaScript style serial number to the the Z3 Controls NetMeter time format
function toZ3Time(time, ybase)
{
    var UTCOffset=parseInt(Date.UTC(ybase, 0, 1)); // number of ms to subtract from JavaScript time
    return (time-UTCOffset)/1000;
}
```

5.3 Command Limitations

Due to the limited buffering capacity inside the Gateway processor, HTTP 'GET' requests are limited to a maximum of 80 characters. This should not be a concern regarding query requests because they are designed to be brief. In cases where more data is required to be sent to the server, the HTTP 'POST' method can be used.

5.4 Return Data

When the HTTP 'GET' or 'POST' request is processed by the Gateway, it will typically respond with a JSON formatted string unless otherwise noted.

JSON (JavaScript Object Notation) is an open source lightweight data-interchange format. It is easy for programmers to read and write and also easy for machines to parse and generate. JSON is a text format that is entirely language independent and is supported by most common programming/scripting languages such as the C-family of languages (C, C + +, C#), Java, JavaScript, Perl, Python, PHP, Tcl, Matlab, and many others.

More information about JSON is available online at <http://www.json.org/>

An example of a JSON response from the NetMeter:

```
{
  "cmd": "sdata.json",
  "arg_s": 0,
  "time": 73069500,
  "ybase": 2010,
  "arg_m": "1",
  "irms": [93, 93, 93, 93],
  "vrms": [223, 223, 223],
  "watt": [20648, 20647, 20637],
  "va": [20649, 20648, 20638],
  "var_": [3, 12, 12],
  "power": 61931,
  "angle": [0, 0, 0],
  "period": 16660,
  "freq": 60,
  "energy": 2876333774
}
```

This is a response containing real-time electrical information from the NetMeter. It is a data object that contains a series of data elements such as “cmd”, “arg_s” and so forth and may be set in any order. The values for these elements can be strings, numbers, arrays, or even hierarchical data objects.

The JSON responses that are described in this document may contain additional data elements to those listed above, depending upon the firmware revision. These can be ignored.

5.5 Modeless Operation

API commands are designed to be as modeless as possible. That is, the API assumes that multiple clients wish to access data simultaneously. Consequently, each command is self-contained and does not rely on the state or “mode” of the gateway/sensor from a previous command.

6 API Command Reference

There are two main classes of commands defined by the Gateway:

1. **Gateway specific commands:** these commands are processed by the Gateway itself. No communication with the sensor(s) is required.
2. **Sensor specific commands:** these commands are sent directly to the sensor(s) for processing.

6.1 Gateway Specific Commands

Gateway specific commands are commands that are processed by the Gateway itself. Gateway commands lead with the letter 'g'.

6.1.1 Gateway Network Configuration: gnconfig.json

Returns the gateway Ethernet/WiFi network configuration.

Command	gnconfig.json	
Parameters	ledfast	This optional parameter causes the heartbeat LED to blink faster for 2 seconds
When no parameters are used, a report of the gateway is generated (see example below)		

Example for using the gnconfig.json query:

Example Query:

`http://netmeter/gnconfig.json`

Example Response:

```
{
  "model": "NetMeter-3P-600-100",
  "hwver": "1.02",
  "fwver": "1.0.0",
  "fwdate": "2012-04-14",
  "fwbuild": "0178",
  "mac": "00:04:A3:00:00:00",
  "ip": "192.168.1.35",
  "mask": "255.255.255.0",
  "gateway": "192.168.1.1",
  "dns1": "8.8.8.8",
  "dns2": "192.168.1.1",
  "hostname": "NETMETER2",
  "dhcp": "0",
  "wtype": "0",
  "remip": "192.168.1.88",
  "remmac": "20:CF:30:AE:AD:1E",
  "auth": "0x80",
  "dflt": {
    "ip": "192.168.2.200",
    "mask": "255.255.255.0",
    "gateway": "192.168.2.75",
    "dns1": "192.168.2.75",
    "dns2": "0.0.0.0",
    "hostname": "NETMETER",
    "dhcp": "1"
  },
  "nvmem": {
    "ip": "192.168.1.35",
    "mask": "",
    "gateway": "192.168.1.1",
    "dns1": "192.168.1.1",
    "dns2": "8.8.8.8",
    "hostname": "NETMETER2",
    "dhcp": "0"
  }
}
```

This data structure is described in Table 2.

Table 2: Description of the Data Structure Returned by the "gnconfig.json" Query

Mnemonic	Type	Description
model	STR	Hardware model number of the Z3 Controls Gateway
hwver	STR	Gateway hardware version
fwver	STR	Gateway firmware version
fwdate	STR	Gateway firmware build date
fwbuild	STR	Gateway firmware build number
mac	MAC	Gateway mac address
ip	IP	Current IP address

Mnemonic	Type	Description
mask	IP	Current IP mask
gateway	IP	Current IP address of the internet gateway (this is separate from the Z3 "Gateway")
dns1	IP	Current IP address for the first domain name server used by the NetMeter
dns2	IP	Current IP address for the second domain name server used by the NetMeter
hostname	STR	The NETBIOS name used to identify the NetMeter on the network
dhcp	BOOL	DHCP (Dynamic Host Configuration Protocol) enabled (= 1) or disabled (= 0)
wtype	U8	Wireless Network Type: "0" indicates wired Ethernet
remip	IP	IP address of the remote client
remmac	MAC	MAC address of the remote client
auth	U8	Authentication level of the client: "0x80" indicates admin level authentication.
dflt.ip	IP	Factory default IP address
dflt.mask	IP	Factory default IP mask
dflt.gateway	IP	Factory default IP address of the internet gateway
dflt.dns1	IP	Factory default IP address for the first domain name server used by the NetMeter
dflt.dns2	IP	Factory default IP address for the second domain name server used by the NetMeter
dflt.hostname	STR	The factory default NETBIOS name used to identify the NetMeter on the network
dflt.dhcp	BOOL	Factory default DHCP setting
nvmem.ip	IP	Power-on default IP address
nvmem.mask	IP	Power-on default IP mask

Mnemonic	Type	Description
nvmem.gateway	IP	Power-on default IP address of the internet gateway
nvmem.dns1	IP	Power-on default IP address for the first domain name server used by the NetMeter
nvmem.dns2	IP	Power-on default IP address for the second domain name server used by the NetMeter
nvmem.hostname	STR	The power-on default NETBIOS name used to identify the NetMeter on the network
nvmem.dhcp	BOOL	Power-on default DHCP setting

6.2 Energy Sensor Specific Commands

This section documents commands that are specific to energy Sensor class devices.

6.2.1 Sensor Information: `sinfo.json`

The sensor information command (`sinfo.json`) is designed to report on the configuration and operation of the energy monitoring subsystem of the NetMeter. It contains information that typically doesn't change over time as the NetMeter collects data. For example, the type of CT in use, or the name and description of the sensor.

Command	<code>sinfo.json</code>	
Parameters	<code>id</code>	<p>This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.</p> <p>Example: <code>sinfo.json?id=abc123</code> will return a data structure with the data member <code>arg_id</code> set to <code>abc123</code>.</p>
	<code>logenable</code>	<p>This optional parameter is used to enable/disable the data logger.</p> <p><code>sinfo.json?logenable=on</code> will enable the data logger</p> <p><code>sinfo.json?logenable=off</code> will disable the data logger</p>
	<code>logclr</code>	<p>This optional parameter is used to clear all data in the data logger.</p> <p><code>sinfo.json?logclr=IAMSURE</code> will clear the data</p>

Example of the `sinfo.json` query:

Example Query:

`http://netmeter/sinfo.json`

Example Response:

```
{
  "model": "NetMeter-3P-600-100",
  "hwver": "1.02",
  "fwver": "1.0.0",
  "fwdate": "2012-04-15",
  "fwbuild": "0180",
  "phmax": "3",
  "ybase": "2010",
  "sensen": "1",
  "time": "72393831",
  "vpsumv": "23600",
  "batok": "1",
  "f0sp": "1",
  "f0asp": "15",
  "f0bsp": "300",
  "f1sp": "60",
  "f2sp": "3600",
  "feature": "78",
  "vdiv": "5.925293e+03",
  "idiv": "1.185651e+07",
  "pdiv": "8.374847e+03",
  "ediv": "1.919040e+03",
  "vmul": [1.687680154163e-04, 1.687680154163e-04, 1.687680154163e-04],
  "imul": [5.065575717887e-05, 5.065575717887e-05, 5.065575717887e-05, 5.065575717887e-05],
  "pmul": 7.171481096397e-02,
  "emul": 2.608076793215e-03,
  "fmul": 2.560000000000e+05,
  "logenable": "0",
  "label": "",
  "desc": "",
  "ctv": "0.333",
  "cta": "200",
  "phact": "3",
  "red": "2500",
  "yel": "1500",
  "revnrg": "0",
  "angsel": "",
  "vxg": "1",
  "logmask0": "",
  "logmask1": "",
  "logperiod": "",
  "logauto": "",
  "vx0": "1",
  "vx1": "1",
  "vx2": "1",
  "ix0": "1",
  "ix1": "1",
  "ix2": "1",
  "ix3": "1",
  "dccpl": "0",
  "senab": "1"
}
```

The returned data structure is described in Table 3

Table 3: Description of the Data Structure Returned by the "sinfo.json" Query

Mnemonic	Type	Description
model	STR	Hardware model number of the Z3 Controls sensor
hwver	STR	Sensor hardware version
fwver	STR	Sensor firmware version
fwdate	STR	Sensor firmware build date
fwbuild	STR	Sensor firmware build number

Mnemonic	Type	Description
phmax	U8	Maximum number of current/voltage phases (3 for the NetMeter-3P)
ybase	U16	Year base: January 1 at midnight (hour 0) of ybase is the zero point in time for all time values. See Section 5.2 for a description of how time stamp values are formulated.
sensen	BOOL	Sensor data storage enable: 0 when disabled, 1 when the sensor is enabled to store data history
time	U32	The current sensor time based on the time the query was transmitted. See Section 5.2 for a description of how time stamp values are formulated.
vpsumv	U16	NetMeter power supply voltage (in millivolts): this is the low voltage power supply for the NetMeter, not the high voltage sensor inputs. This has roughly +/- 10% accuracy.
batok	STR	Real time clock backup battery status: 0 when not present or needs replacement, 1 indicates that it is OK
f0sp	U16	Power Data FIFO0 sample rate in seconds: this is fixed at a value of "1" for the NetMeter-3P
f0asp	U16	Power Data FIFO0A sample rate in seconds: this is fixed at a value of "15" for the NetMeter-3P
f0bsp	U16	Power Data FIFO0A sample rate in seconds: this is fixed at a value of "300" for the NetMeter-3P
f1sp	U16	Power Data FIFO1 sample rate in seconds: this is fixed at a value of "60" for the NetMeter-3P
f2sp	U16	Power Data FIFO1 sample rate in seconds: this is fixed at a value of "3600" for the NetMeter-3P
feature	U16	Sensor feature set: a value of "78" denotes the standard NetMeter-3P
vmul[0-2]	F64	Voltage scale factor used to scale raw voltage data from the sdata.json query. There is one scale factor per voltage input. See the sdata.json query for more details.
imul[0-3]	F64	Current scale factor used to scale raw current data from the sdata.json query. There is one scale factor per current input. See the sdata.json query for more details.
pmul	F64	Power scale factor used to scale raw power data from the sdata.json query. The same pmul value is applied to all raw power data. See the sdata.json query for more details.
emul	F64	Energy scale factor used to scale raw cumulative energy data from the sdata.json query. The same emul value is applied to all cumulative energy data. See the sdata.json query for more details.
fmul	F64	Frequency scale factor used to scale raw period information so that the line frequency may be calculated. See the sdata.json query for more details.

Mnemonic	Type	Description
logenable	BOOL	Datalog enable: 0 when the data logger is disabled, 1 when it is enabled
label	STR (W)	The user defined label for the NetMeter sensor
desc	STR (W)	The user defined description of the NetMeter sensor
ctv	F32 (W)	Current transformer voltage output at rated current. The combination of ctv and cta determine the transfer function of the current transformers. This is used to scale the current, power, and energy data.
cta	F32 (W)	Current transformer rated current
phact	U8 (W)	Actual number of valid phases: may be less than or equal to phmax
red	U32 (W)	The power (in watts) at which the graphical interface will indicate red when plotting power. Values below this level will transition to yellow.
yel	U32 (W)	The power (in watts) at which the graphical interface will indicate yellow when plotting power. Values below this level will transition to green.
revnrg	BOOL (W)	Reverse energy: 1 indicates that energy flow may be in either a positive or negative direction. When 0 is indicated, the NetMeter will capture the absolute value of energy flow.
angsel	STR (W)	Phase angle monitor select.
vxg	F32 (W)	Global voltage multiplier: voltage values are to be multiplied by this number. A value of 1.0 is normal. However, non unity values are used to indicate the presence of transformers in the voltage path.
logmask0	U32 (W)	A bitmask of what parameters are to be recorded by the data logger (part 1)
logmask1	U32 (W)	A bitmask of what parameters are to be recorded by the data logger (part 2)
logperiod	U16 (W)	Log interval (in seconds)
logauto	BOOL (W)	Data logger auto start: 1 indicates that logging should automatically restart whenever the NetMeter power cycles. 0 indicates that data logging will be disabled after a power failure.
vx0	F32 (W)	Voltage scale channel 0 (Phase A voltage input)
vx1	F32 (W)	Voltage scale channel 1 (Phase B voltage input)
vx2	F32 (W)	Voltage scale channel 2 (Phase C voltage input)
ix0	F32 (W)	Current scale channel 0 (Phase A CT input)
ix1	F32 (W)	Current scale channel 1 (Phase B CT input)
ix2	F32 (W)	Current scale channel 2 (Phase C CT input)
ix3	F32 (W)	Current scale channel 3 (Phase D CT input)

Mnemonic	Type	Description
dccpl	BOOL (W)	DC Couple: reports if the voltage/current inputs are set for DC coupling (1) or AC coupling (0)
senab	BOOL (W)	Power-on sensor enable: determines the state of the sensor enable bit after a power-on reset.

6.2.2 Sensor Data Query: sdata.json

The query “sdata.json” is used to obtain sensor data, either real time or historical. It is a multimode command with many parameters. For clarity, each mode will be documented separately below.

There are 2 categories of data available from the NetMeter-3P:

1. Data captured directly from the sensor subsystem in real-time
2. Data that has been stored in on-board memory.

There are a number of data storage arrays in the NetMeter-3P:

- FIFO0 has volatile storage for 60 seconds worth of total power consumption, data sampled every second
- FIFO0A has volatile storage for 2 hours worth of total power consumption, data sampled every 15 seconds
- FIFO0A has volatile storage for 2 days worth of total power consumption, data sampled at 5 minute intervals
- FIFO1 has non-volatile storage for over 2 years worth of total power consumption, data sampled at one minute intervals
- FIFO1H has non-volatile storage for over 2 years worth of total accumulated energy consumption, data sampled at one hour intervals
- FIFO2 has non-volatile storage for over 10 years worth of total accumulated energy consumption, data sampled at one hour intervals in addition to system power events
- The data logger can store up to 47 different parameters at a selectable interval.

In most cases, data contained in data queries (voltage/current/power/energy etc.) can be sent as integer numbers that must be scaled in order to convert them to standard units. This is referred to as raw data. This approach has two primary advantages:

- Integer data is typically smaller to transmit as text compared to floating point numbers
- The integer data is native to the data acquisition and digital signal processing subsystem. Consequently, the highest precision is maintained during data transmission.

Alternately, many commands have an option for the data to be scaled inside the NetMeter. This is referred to as scaled data. With scaled data, some small loss of precision is seen, or an expansion of the data size can result when high floating point precision is specified.

When the raw data mode is used, the process of obtaining data from the NetMeter proceeds as follows:

1. Execute the `sinfo.json` query (Section 6.2.1). This is required only once for a series of data queries. It provides the scale values that are used to convert the raw sensor data into standard values such as volts, amps, watts, and kilowatt-hours.
2. Execute multiple `sdata.json` queries (Section 6.2.2) and scale it according to the results of the `sinfo.json` query.

For raw data, values are scaled according to Table 4 below:

Table 4: Scaling of Raw Sensor Data

Electrical Parameter	Units	Scale Factor (from <code>sinfo.json</code>)	Raw Data (from <code>sdata.json</code>)
Voltage Input A	Volts RMS	<code>vmul[0]</code>	<code>vrms[0]</code>
Voltage Input B	Volts RMS	<code>vmul[1]</code>	<code>vrms[1]</code>
Voltage Input C	Volts RMS	<code>vmul[2]</code>	<code>vrms[2]</code>
Current Input A	Amps RMS	<code>imul[0]</code>	<code>irms[0]</code>
Current Input B	Amps RMS	<code>imul[1]</code>	<code>irms[1]</code>
Current Input C	Amps RMS	<code>imul[2]</code>	<code>irms[2]</code>
Current Input D	Amps RMS	<code>imul[3]</code>	<code>irms[3]</code>
Actual Power Input A	Watts	<code>pmul</code>	<code>watt[0]</code>
Actual Power Input B	Watts	<code>pmul</code>	<code>watt[1]</code>
Actual Power Input C	Watts	<code>pmul</code>	<code>watt[2]</code>
Total Actual Power (A + B + C)	Watts	<code>pmul</code>	<code>power</code>
Apparent Power (VA) Input A	Watts	<code>pmul</code>	<code>va[0]</code>
Apparent Power (VA) Input B	Watts	<code>pmul</code>	<code>va[1]</code>
Apparent Power (VA) Input C	Watts	<code>pmul</code>	<code>va[2]</code>
Reactive Power (VAR) Input A	Watts	<code>pmul</code>	<code>var_[0]</code>
Reactive Power (VAR) Input B	Watts	<code>pmul</code>	<code>var_[1]</code>
Reactive Power (VAR) Input C	Watts	<code>pmul</code>	<code>var_[2]</code>
Active Energy Input A	Watt-hours ¹	<code>emul</code>	<code>watthr[0]</code>
Active Energy Input B	Watt-hours	<code>emul</code>	<code>watthr[1]</code>
Active Energy Input C	Watt-hours	<code>emul</code>	<code>watthr[2]</code>
Active Energy Total	Watt-hours	<code>emul</code>	<code>energy</code>

¹ For kilowatt-hours instead of watt-hours, divide by 1,000.

Electrical Parameter	Units	Scale Factor (from sinfo.json)	Raw Data (from sdata.json)
Apparent Energy Input A	Watt-hours	emul	vahr[0]
Apparent Energy Input B	Watt-hours	emul	vahr[1]
Apparent Energy Input C	Watt-hours	emul	vahr[2]
Reactive Energy Input A	Watt-hours	emul	varhr[0]
Reactive Energy Input B	Watt-hours	emul	varhr[1]
Reactive Energy Input C	Watt-hours	emul	varhr[2]
Fundamental Active Energy Input A	Watt-hours	emul	fwatthr[0]
Fundamental Active Energy Input B	Watt-hours	emul	fwatthr[1]
Fundamental Active Energy Input C	Watt-hours	emul	fwatthr[2]
Fundamental Reactive Energy Input A	Watt-hours	emul	fvarhr[0]
Fundamental Reactive Energy Input B	Watt-hours	emul	fvarhr[1]
Fundamental Reactive Energy Input C	Watt-hours	emul	fvarhr[2]
Line Period (See Text)	Seconds	1/fmul	period
Phase Angle Input A		See Text	angle[0]
Phase Angle Input B		See Text	angle[0]
Phase Angle Input C		See Text	angle[0]

Phase angle for each channel is calculated as:

$$\text{Phase Angle Input } x \text{ (Degrees)} = 360 * \text{angle}[x] / \text{period}$$

Line frequency can be calculated as:

$$\text{Line Frequency (Hz)} = \text{fmul} / \text{period}.$$

6.2.2.1 Sensor Data Query: Realtime Data (Mode 1)

The Mode 1 data query returns all real-time non-cumulative data:

- Voltage (each phase)
- Current (each phase)
- Power (each phase and total)
- VA, VAR (each phase)
- Phase angle (each phase)
- Line period (Phase A)

Command	sdata.json?m=1	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.
	s (U8)	This is an optional integer parameter to specify that data should be scaled to standard units. The number specified sets the number of decimal places of precision. In this mode, the line frequency (in Hz) is calculated (called "freq" in the data structure) in addition to the line period (in milliseconds). When the "s" parameter is not specified, all data is sent as raw data that must be scaled according to Table 4.

Mode 1 Example (raw data):

Example Query:

`http://netmeter/sdata.json?m=1&id=mode%20example`

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 72642917,
  "arg_id": "mode 1 example",
  "arg_m": "1",
  "irms": [3647962, 3648249, 3647878, 3647897],
  "vrms": [1317966, 1318355, 1318349],
  "watt": [573179, 573379, 573327],
  "va": [573133, 573337, 573281],
  "var_": [142, 361, 306],
  "power": 1719885,
  "energy": 83859824,
  "angle": [4264, 4264, 4264],
  "period": 4264
}
```

The raw data is scaled according to Table 4.

In the next example, the data is scaled by the NetMeter instead of having to post process it.

Example (scaled data using the "s" parameter):

Example Query:

`http://netmeter/sdata.json?m=1&s=1`

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 72666770,
  "arg_s": 1,
  "arg_m": "1",
  "irms": [92.4, 92.4, 92.4, 92.4],
  "vrms": [222.6, 222.6, 222.6],
  "watt": [20574.3, 20578.6, 20573.0],
  "va": [20573.2, 20577.9, 20573.3],
  "var_": [3.3, 11.9, 10.6],
  "power": 61725.9,
  "angle": [0.0, 0.1, 0.0],
  "period": 16668.0,
  "freq": 59.0,
  "energy": 239598165
}
```


6.2.2.2 Sensor Data Query: Realtime Data (Mode 2)

The Mode 2 data query returns all real-time cumulative energy data:

- Total Active Energy (each phase an total)
- Fundamental Active Energy (each phase)
- Total Apparent Energy (each phase)
- Total Reactive Energy (each phase)
- Fundamental Reactive Energy (each phase)

Command	sdata.json?m=2	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.
	s (U8)	When this optional integer parameter is specified, the scale factor for energy ("emul") is included with the data set. Consequently, all energy data may be scaled without having to execute the sinfo.json query.

Mode 2 Example:

Example Query:

```
http://netmeter/sdata.json?m=2&id=example
```

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 72643879,
  "arg_id": "example",
  "arg_m": "2",
  "watthr": [26172455, 34932080, 29079056],
  "vahr": [26172454, 34932067, 29079053],
  "varhr": [6902, 23443, 17103],
  "fwatthr": [26115089, 34855599, 29015247],
  "fvarhr": [5784, 22485, 16318],
  "energy": 90183591
}
```

6.2.2.3 Sensor Data Query: Realtime Data (Mode 3)

The Mode 3 data query combines all the result from mode 1 and mode 2 into a single query. See mode 1 and mode 2 description for more details.

Command	sdata.json?m=3	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.
	s (U8)	<p>This is an optional integer parameter to specify that data should be scaled to standard units. The number specified sets the number of decimal places of precision. The "s" parameter affects "irms", "vrms", "watt", "va", "var_", "power", "angle", and "period". It does not apply to any energy data. However, when the "s" parameter is specified the scale factor for energy ("emul") is included with the data set. Consequently, all energy data may be scaled without having to execute the sinfo.json query.</p> <p>When the "s" parameter is not specified, all data is sent as raw data that must be scaled according to Table 4 using scale factors from the sinfo.json query.</p>

Example Query for Mode 3:

Query:

```
http://netmeter/sdata.json?m=3&id=mode%20%20example
```

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 72667286,
  "arg_id": "mode 3 example",
  "arg_m": "3",
  "irms": [3643245, 3642891, 3641927, 3641907],
  "vrms": [1316413, 1316510, 1316286],
  "watt": [571707, 571678, 571448],
  "va": [571752, 571736, 571474],
  "var_": [82, 280, 303],
  "power": 1714833,
  "angle": [4266, 4266, 4266],
  "period": 4266,
  "energy": 242974503,
  "watthr": [77100820, 85869515, 80004168],
  "vahr": [77100824, 85869513, 80004177],
  "varhr": [15112, 53090, 43792],
  "fwatthr": [76924653, 85674168, 79821537],
  "fvarhr": [12457, 50902, 41788],
  "energy": 242974503
}
```

Or with the "s" option:

Example Query for Mode 3 with the “s” option:

```
http://netmeter/sdata.json?m=3&s=2
```

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 72667938,
  "arg_s": 2,
  "arg_m": "3",
  "irms": [92.56, 92.56, 92.55, 92.55],
  "vrms": [222.84, 222.88, 222.89],
  "watt": [20626.29, 20631.85, 20629.52],
  "va": [20624.53, 20630.27, 20628.37],
  "var_": [3.59, 11.51, 10.18],
  "power": 61887.66,
  "angle": [0.00, 0.00, 0.00],
  "period": 16664.06,
  "freq": 60.00,
  "energy": 247249153,
  "watthr": [78525712, 87294586, 81428855],
  "vahr": [78525716, 87294584, 81428865],
  "varhr": [15202, 53867, 44501],
  "fwatthr": [78346296, 87095989, 81242976],
  "fvarhr": [12566, 51643, 42462],
  "energy": 247249153,
  "emul": 2.608076793215e-03
}
```

6.2.2.4 Sensor Data Query: Recent History (Modes “f0”, “f0a”, “f0b”)

The internal data structure called FIFO0 provides volatile storage for recent power use:

- The last 1 minute at 1 second resolution (m = f0)
- The last 2 hours at 15 sec resolution (m = f0a)
- The last 2 days at 5 minute resolution (m = f0b)

The intended use of this data is for real-time displays and contains only power data across all phases combined. For more historical power details, the datalog should be used. For long term non-volatile data and cumulative energy, use FIFO1 (1 minute resolution) and FIFO2 (1 hour resolution).

Table 5: FIFO0 Query Options

Command	sdata.json?m=f0 (The last 1 minute at 1 second resolution) sdata.json?m=f0a (The last 2 hours at 15 sec resolution) sdata.json?m=f0b (The last 2 days at 5 minute resolution)
Parameters	id (STR) This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.

s (U8)

This is an optional integer parameter to specify that data should be scaled to standard units of Watts.

When the “s” parameter is not specified, all data is sent as raw data that must be scaled according to the “pmul” value returned by the query.

Example Query for FIFO0:

Query:

```
http://netmeter/sdata.json?m=f0
```

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 72893616,
  "ybase": 2010,
  "arg_m": "f0",
  "sp": 1,
  "power": [1751120, 1753022, 1755813, 1756253, 1756662, 1747054, 1751166, 1754151, 1750494, 1753005,
1750994, 1753979, 1754918, 1754831, 1750064, 1754329, 1751265, 1752965, 1749748, 1745752, 1748838, 1750035,
1748797, 1751349, 1746342, 1749593, 1749790, 1750048, 1749791, 1749447, 1749825, 1749206, 1749898, 1748355,
1744402, 1747409, 1746862, 1744491, 1748539, 1747711, 1747437, 1749863, 1751103, 1748343, 1749903, 1751892,
1747665, 1746274, 1747463, 1749634, 1748917, 1747340, 1747605, 1748856, 1748453, 1747644, 1747686, 1748976,
1749631, 1757339, 1755828],
  "pmul": 3.585740548199e-02
}
```

The resulting data in “power” is scaled using “pmul” to convert it to Watts.

Or with the “s” option, the data is scaled by the NetMeter as in the Example below:

Query:

```
http://netmeter/sdata.json?m=f0&s=2
```

Example Response:

```
{
  "cmd": "sdata.json",
  "arg_s": 2,
  "time": 72893691,
  "ybase": 2010,
  "arg_m": "f0",
  "sp": 1,
  "power": [61810.64, 61722.32, 61592.62, 61641.21, 61723.65, 61760.44, 61905.16, 61690.08, 61695.03,
61656.84, 61862.52, 61758.93, 61841.01, 61738.81, 61842.05, 61804.65, 61804.15, 61836.35, 62092.26,
61933.95, 61945.96, 62028.40, 62039.44, 61784.60, 61933.81, 61848.83, 61805.87, 62033.99, 61943.38,
61956.40, 61805.26, 62019.94, 62031.05, 61848.04, 61734.12, 61826.41, 61804.29, 61701.20, 61754.84,
61937.89, 61905.05, 61673.16, 61712.14, 61740.54, 61823.19, 61659.35, 61834.70, 61842.91, 61772.84,
62056.22, 61996.74, 61949.44, 62038.73, 61939.51, 61794.43, 62059.56, 62117.97, 62418.74, 63082.39,
63001.64, 63037.07]
}
```

Note that the first data element in power (power[0]) is the most recent as of “time”. Subsequent data is for older time stamps (the data is in reverse chronological order).

In the case of “m=f0”, the data ages by 1 second (as specified by the “sp” value) throughout the array.

The f0a and f0b commands are similar except that the quantity of data and sample spacing differs: 15 second for f0a and 5 min for f0b. The time delta between samples is (in seconds) can be obtained from the "sp" value.

Table 6: Description of the Data Structure Returned by the "sdata.json" Query for FIFO0

Mnemonic	Type	Description
cmd	STR	Indicates the sdata.json query
arg_id	STR	Value set by the "id" parameter in the query
arg_s	U8	Value set by the "s" parameter in the query
arg_m	STR	Mode of the sdata.json query: "f0", "f0a", or "f0b"
sp	U16	Sample period of the data in "power". This can be used as a delta-t to determine the time stamp of all data in "power"
time	U32	The current sensor time as of the time the query was being transmitted. The time of the power[0] array element.
ybase	U16	Year base: See Section 5.2.
power	U32 or F32	An array containing either raw power values (U32) or scaled power values (F32) in reverse chronological order.
pmul	F64	Power scale factor used to scale raw power data for when "s" is not specified for the query.

6.2.2.5 Power Use History: 1 Minute Resolution (Mode "f1t")

The one minute resolution power use data can be accessed using the "f1t" mode of the sdata.json query. The query returns the average power for each minute of a specified period. If the NetMeter was powered off, and thus not recording any data, then data will not be returned for these periods of time.

The query options are given in Table 7

Table 7: Query Options for sdata.json?m=f1t

Command	sdata.json?m=f1t	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.
	s (U8)	This is an optional integer parameter to specify that data should be scaled to standard units of Watts. When the "s" parameter is not specified, all data is sent as raw data that must be scaled according to the "pmul" value returned by the query.
	t (U32)	Start time: the returned data will begin at time "t"

te (u32)	End time: the returned data will end at time "te". When no "te" or "d" argument is specified or "te" > current time, the end time will be the current time.
<hr/>	
d (U32)	Duration of data range: as an alternate to specifying "te", a duration may be specified instead. The end time of the data range will be "t" + "d". When no "te" or "d" argument is specified or "t" + "d" > current time, the end time will be the current time.

Example Query where the time range is specified:

Query:

<http://netmeter/sdata.json?m=f1t&t=73047000&te=73064340>

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 73064682,
  "ybase": 2010,
  "arg_m": "f1t",
  "arg_t": 73047000,
  "arg_te": 73064340,
  "pmul": 3.585740548199e-02,
  "power": [
    [73047000, [1738752, 1746944, 1760256, 1755136, 1757184, 1755136, 1755136, 1754112, 1744896, 1744896]],
    [73047600, [1743872, 1741824, 1737728, 1732608, 1735680, 1733632, 1733632, 1730560, 1722368, 1735680,
    1753088, 1755136, 1764352, 1727488, 1731584, 1730560, 1732608, 1732608, 1730560, 1729536, 1734656,
    1733632, 1733632, 1741824, 1712128, 1726464, 1744896, 1740800, 1740800, 1738752, 1740800, 1737728,
    1735680, 1738752, 1736704, 1738752, 1737728, 1739776, 1737728, 1731584, 1731584, 1733632, 1736704,
    1736704, 1736704, 1738752, 1739776, 1740800, 1737728, 1738752, 1755136, 1758208, 1760256, 1758208,
    1759232, 1759232, 1756160, 1738752, 1734656, 1738752, 1733632, 1734656, 1733632, 1731584, 1733632,
    1735680, 1734656, 1736704, 1735680, 1738752, 1746944, 1756160, 1743872, 1735680, 1737728, 1740800,
    1740800, 1738752, 1737728, 1734656, 1736704, 1739776, 1743872, 1745920, 1742848, 1741824,
    1744896, 1736704, 1748992, 1746944, 1747968, 1744896, 1742848, 1742848, 1743872, 1744896, 1746944,
    1759232, 1769472, 1768448, 1767424, 1764352, 1766400, 1764352, 1759232, 1752064, 1738752, 1736704,
    1735680, 1735680, 1737728, 1737728, 1741824, 1741824, 1739776, 1738752, 1742848, 1745920, 1752064]],
    [73054800, [1765376, 1748992, 1744896, 1746944, 1742848, 1743872, 1744896, 1746944, 1745920, 1738752,
    1740800, 1738752, 1739776, 1739776, 1740800, 1740800, 1740800, 1743872, 1745920, 1745920, 1746944,
    1746944, 1744896, 1743872, 1740800, 1739776, 1742848, 1763328, 1764352, 1765376, 1763328, 1762304,
    1765376, 1761280, 1739776, 1736704, 1740800, 1741824, 1743872, 1744896, 1744896, 1743872, 1748992,
    1751040, 1752064, 1751040, 1748992, 1763328, 1769472, 1751040, 1748992, 1750016, 1746944, 1750016,
    1751040, 1754112, 1756160, 1756160, 1754112, 1757184, 1756160, 1756160, 1755136, 1755136, 1753088,
    1752064, 1751040, 1754112, 1754112, 1756160, 1758208, 1757184, 1770496, 1774592, 1775616, 1776640,
    1775616, 1773568, 1769472, 1747968, 1734656, 1723392, 1715200, 1732608, 1731584, 1744896, 1756160,
    1756160, 1754112, 1748992, 1737728, 1728512, 1732608, 1732608, 1731584, 1732608, 1729536, 1727488,
    1731584, 1731584, 1733632, 1755136, 1745920, 1716224, 1725440, 1728512, 1731584, 1734656, 1731584,
    1732608, 1725440, 1721344, 1724416, 1725440, 1726464, 1729536, 1730560, 1730560, 1729536, 1729536]],
    [73062000, [1728512, 1730560, 1732608, 1730560, 1729536, 1728512, 1731584, 1753088, 1752064, 1747968,
    1753088, 1753088, 1750016, 1748992, 1741824, 1725440, 1740800, 1729536, 1727488, 1725440, 1730560,
    1737728, 1739776, 1738752, 1737728, 1735680, 1736704, 1740800, 1737728, 1758208, 1747968, 1732608,
    1748992, 1754112, 1754112, 1746944, 1746944, 1741824, 1735680]],
    [73064340, [1750016]]]
  ]
}
```

Or with the "s" option, the data is scaled by the NetMeter as in the next example:

Query:

<http://netmeter/sdata.json?m=f1t&t=73047000&te=73064340&s=0>

Example Response:

```
{
  "cmd": "sdata.json",
  "arg_s": 0,
  "time": 73064724,
  "ybase": 2010,
  "arg_m": "f1t",
  "arg_t": 73047000,
  "arg_te": 73064340,
  "power": [
    [73047000, [62347, 62641, 63118, 62935, 63008, 62935, 62935, 62898, 62567, 62567]],
    [73047600, [62531, 62457, 62310, 62127, 62237, 62164, 62164, 62053, 61760, 62237, 62861, 62935, 63265, 61943,
    62090, 62053, 62127, 62127, 62053, 62017, 62200, 62164, 62164, 62457, 61392, 61907, 62567, 62421, 62421, 62347,
    62421, 62310, 62237, 62347, 62274, 62347, 62310, 62384, 62310, 62090, 62090, 62164, 62274, 62274, 62274, 62347,
    62384, 62421, 62310, 62347, 62935, 63045, 63118, 63045, 63081, 63081, 62971, 62347, 62200, 62347, 62164, 62200,
    62164, 62090, 62164, 62237, 62200, 62274, 62237, 62347, 62641, 62971, 62531, 62237, 62310, 62421, 62421, 62347,
    62310, 62310, 62200, 62274, 62384, 62531, 62604, 62494, 62457, 62567, 62274, 62714, 62641, 62678, 62567, 62494,
    62494, 62531, 62567, 62641, 63081, 63449, 63412, 63375, 63265, 63339, 63265, 63081, 62824, 62347, 62274, 62237,
    62237, 62310, 62310, 62457, 62457, 62384, 62347, 62494, 62604, 62824]],
    [73054800, [63302, 62714, 62567, 62641, 62494, 62531, 62567, 62641, 62604, 62347, 62421, 62347, 62384, 62384,
    62421, 62421, 62421, 62531, 62604, 62604, 62641, 62641, 62567, 62531, 62421, 62384, 62494, 63228, 63265, 63302,
    63228, 63192, 63302, 63155, 62384, 62274, 62421, 62457, 62531, 62567, 62567, 62531, 62714, 62788, 62824, 62788,
    62714, 63228, 63449, 62788, 62714, 62751, 62641, 62751, 62788, 62898, 62971, 62971, 62898, 63008, 62971, 62971,
    62935, 62935, 62861, 62824, 62788, 62898, 62898, 62971, 63045, 63008, 63485, 63632, 63669, 63706, 63669, 63596,
    63449, 62678, 62200, 61796, 61503, 62127, 62090, 62567, 62971, 62971, 62898, 62714, 62310, 61980, 62127, 62127,
    62090, 62127, 62017, 61943, 62090, 62090, 62164, 62935, 62604, 61539, 61870, 61870, 61980, 62090, 62200, 62090, 62127,
    61870, 61723, 61833, 61870, 61907, 62017, 62053, 62053, 62017, 62017]],
    [73062000, [61980, 62053, 62127, 62053, 62017, 61980, 62090, 62861, 62824, 62678, 62861, 62861, 62751, 62714,
    62457, 61870, 62421, 62017, 61943, 61870, 62053, 62310, 62384, 62347, 62310, 62237, 62274, 62421, 62310, 63045,
    62678, 62127, 62714, 62898, 62898, 62641, 62641, 62457, 62237]],
    [73064340, [62751]]]
  ]
}
```

The data structure above is detailed in Table 8.

Table 8: Data Structure Returned by the 1 Minute Resolution Power Query (sdata.json?m="f1t")

Mnemonic	Type	Description
cmd	STR	Indicates the sdata.json query
arg_id	STR	Value set by the "id" parameter in the query
arg_s	U8	Value set by the "s" parameter in the query. This will be the number of decimal places of accuracy for power vales.
arg_m	STR	Mode of the sdata.json query: "f1t"
arg_t	U32	Start time: the value specified in the "t" parameter of the query. Determines the desired start time for the range of power data.
arg_te	U32	End time: the value specified in the "te" parameter of the query. Determines the desired end time for the range of power data.
arg_d	U32	Time duration: the value specified in the "d" parameter of the query. Determines the desired period of time (in seconds) for the power data range. For example, "d=3600" specifies 1 hour of data (60 data samples, assuming that the NetMeter has the data).
time	U32	The current sensor time as of the time the query was being transmitted.
ybase	U16	Year base: See Section 5.2.

Mnemonic	Type	Description
power	U32 or F32	An array containing arrays of time stamped power values, either raw power values (U32) or scaled power values (F32), depending upon the "s" parameter. See the explanation below for further details.
pmul	F64	Power scale factor used to scale raw power data for when "s" is not specified for the query.

The construction of the "power" member of the query results requires further explanation:

- The "power" member of the query results is in an array containing an arbitrary number of arrays that are constructed as:

[TIMESTAMP_i, [POWER₀, POWER₁, POWER₂, ... , POWER_{N-1}]

- Each array in "power" contains 2 elements:
 - A time stamp (TIMESTAMP_i): addressed as "power[i][0]"
 - An array of power values (POWER₀, ... , POWER_{N-1}): addressed as "power[i][1][j]" where N is up to 120 (2 hours worth).
- The array of power values is a sequence of power values spaced by exactly 60 seconds.

The data structure is composed this way in order to conserve space. It does so by negating the requirement for a timestamp to be transmitted for each individual data point. Please note that it is still able to accommodate possible gaps in data during times when the NetMeter is switched off.

Unlike most of the other queries involving time series data, the "f1t" query mode is in chronological order rather than reverse chronological order. Consequently, "power[i][1][0]" is at time "power[i][0]" and "power[i][1][1]" is at time "power[i][0] + 60" and so on.

Consider the following query where a 10 minute duration is specified:

```
http://netmeter/sdata.json?m=f1t&t=72000000&d=600&s=0
```

And the power element of the response (re-formatted for simplicity)

```
"power": [
  [72000000, [100, 200, 300, 400]],
  [72000300, [500, 600, 700, 800, 900, 1000]]
]
```

The response should contain exactly 11 power values for the 10 minute span. However, this example has a discontinuity in the data because the NetMeter was powered off for one of the minutes, so only 10 data points are shown to be available.

The first array has a time stamp of 72000000 which is 2012-04-13 08:00:00 GMT and at that time the power is 100 Watts.

The power increases by 100 Watts for each of the 4 minutes as detailed in Table 9 below. Due to the lack of data for the 5th minute (at 04:00), the 1st array ends and a 2nd time stamped list starting at time 72000300 begins.

Table 9: Breakdown of Example "f1t" Query Results

Time	Power Value
72000000 (2012-04-13 08:00:00)	100
72000060 (2012-04-13 08:01:00)	200
72000120 (2012-04-13 08:02:00)	300
72000180 (2012-04-13 08:03:00)	400
72000240 (2012-04-13 08:04:00)	0 ←There is no data for this time because the NetMeter was powered off at this moment
72000300 (2012-04-13 08:05:00)	500
72000360 (2012-04-13 08:06:00)	600
72000420 (2012-04-13 08:07:00)	700
72000480 (2012-04-13 08:08:00)	800
72000540 (2012-04-13 08:09:00)	900
72000600 (2012-04-13 08:10:00)	1000

6.2.2.6 Accumulated Energy History: 1 Hour Resolution (Mode "f1h")

The "f1h" query produces a report of total cumulative active energy as recorded at the top of each hour. If the NetMeter was powered off at the top of the hour or for an even longer gap of time, then there would be an entry for the start time at which the NetMeter power had resumed.

This query is useful for plotting long term energy use, or to calculate the costs, even where different rates apply at different times of the day (Time of Use rates).

The query options are given in Table 10 below:

Table 10: Query Options for sdata.json?m=f1h

Command	sdata.json?m=f1h	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.
	t (U32)	Optional time filter: only data that is newer than or equal to time "t" will be returned. When "t" is not specified, it will default to 0.
	te (U32)	Optional end time filter: only data that is older than or equal to time "te" will be returned. When omitted, all available hourly data (that meets the criteria of the "t" parameter) up to the current time will be returned.

Example query with no time range specified:

Query:

<http://netmeter/sdata.json?m=f1h>

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 73170457,
  "ybase": 2010,
  "arg_m": "f1h",
  "emul": 2.608076793215e-03,
  "energy": [[73170457, 3541590146], [73170000, 3538603653], [73168800, 3530807122], [73168020, 3525848087], [73166400, 3515480561], [73163820, 3498659888], [73162800, 3491946269], [73162620, 3490740766], [73162260, 3488437717], [73162200, 3488131929], [73159200, 3468339324], [73155900, 3446474415], [73155600, 344508473], [73152000, 3420707246], [73148400, 3396903311], [73144800, 3373282621], [73141200, 3349529914], [73137600, 3325923694], [73134000, 3302267514], [73130400, 3278342749], [73126800, 3254316531], [73123200, 3230488298], [73119600, 3206559571], [73116000, 3182784758], [73112400, 3158991732], [73108800, 3134920354], [73105200, 3111337905], [73101600, 3087696805], [73098000, 3064411630], [73094400, 3040866251], [73090800, 3017107666], [73087200, 2993388825], [73083600, 2970035995], [73080000, 2946284667], [73076400, 2922413663], [73072800, 2898302296], [73069200, 2874336024], [73069080, 2873540559], [73065600, 2850565739], [73064340, 2842146285], [73062000, 2826689526], [73058400, 2802733258], [73054800, 2778682683], [73051200, 2754690995], [73047600, 2730775608], [73044000, 2706740547], [73040400, 2682859905], [73036800, 2659046014], [73033200, 2635188362], [73029600, 2611350758], [73026000, 2587571579], [73022400, 2563570839], [73018800, 2540021006], [73015200, 2516504122], [73014000, 2508714022], [73011600, 2493077077], [73010220, 2484076025], [73008000, 2469666801], [73004400, 2445783098], [73004280, 2444986523], [73003920, 2442692251], [73003380, 2439230105], [73002660, 2434670233], [73001040, 2424062353], [73000800, 2422584177], [72999240, 2412276309], [72999180, 2411971656], [72997200, 2398884641], [72993600, 2374938891], [72990000, 2350939344], [72986400, 2327017870], [72982800, 2303210947], [72979200, 2279401671], [72975600, 2255619238], [72972000, 2231664299], [72968400, 2207783595], [72964800, 2183890699], [72961200, 2159945994], [72957600, 2135906400], [72954000, 2111934567], [72950400, 2087789353], [72946800, 2063618872], [72943200, 2039643520], [72939600, 2015844596], [72937260, 2000427337], [72936360, 1994572062]]
}
```

Each element of the “energy” array contains time (energy[i][0]) and raw energy (energy[i][1]). They are in reverse chronological order, starting with the most recent time and then ‘aging’ throughout the array.

In cases where the NetMeter has been switched off for a period of time, or a series of outages, this will result in intermediate entries into the “energy” array.

When no “te” parameter is specified, the first reported energy accumulation value is the current time/energy value as of the time of the query.

The raw energy values are scaled with “emul” to convert it to a value in Watt-hours. Divide this result by 1000 for kWhr.

The average power between two samples of the “energy” data can be calculated as:

$$\text{Average Power(Watts)} = \text{Delta Energy(Watt-seconds)}/\text{Delta Time(Seconds)}$$
$$= \text{emul} * 3600 * (\text{energy}[i][1] - \text{energy}[i + 1][1]) / (\text{energy}[i][0] - \text{energy}[i + 1][0])$$

Example query with a specified time range:

Query:

<http://netmeter/sdata.json?m=f1h&te=73168800&t=73166400>

Example Response:

```
{
  "cmd": "sdata.json",
  "time": 73170541,
  "ybase": 2010,
  "arg_m": "f1h",
  "arg_t": "73166400",
  "arg_te": 73168800,
  "emul": 2.608076793215e-03,
  "energy": [[ 73168800, 3530807122], [ 73168020, 3525848087], [ 73166400, 3515480561]]
}
```

In this second query, only a small subset of data is returned according to the criteria of the “t” and “te” parameters.

6.2.2.7 Dashboard Statistics: (Mode “stat”)

The “stat” mode of the sdata.json query provides a series of data that is useful for energy dashboards. It provides the data required to display:

- Instantaneous power demand
- Energy used so far this hour, so far today
- Energy used each calendar day for the past week
- Energy used over the past day and on the day before
- Energy used over the past week and in the previous week

Table 11: Dashboard Energy Statistics Query: (Mode “stat”)

Command	sdata.json?m=stat	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier.
	s (U8)	This is an optional integer parameter to specify that the power data should be scaled to standard units of Watts. When the “s” parameter is not specified, power data is sent as raw data that must be scaled according to the “pmul” value returned by the query. Energy data is always in raw format and should be scaled using the “emul” value.

utc (U32) UTC time offset: the time zone offset (in seconds) for the local time zone of the client. For example: Eastern Standard Time has an offset of -5 hours (utc = -18000 seconds) compared to UTC time. (-4 hours during daylight savings time)

“utc” should be aligned to an hour. For time zones that have fractional hour offset, the returned data will be based on the closest hour.

Example Query:

<http://netmeter/sdata.json?m=stat&s=1&utc=-14400&id=MyID>

Example Response:

```
{
  "cmd": "sdata.json",
  "arg_s": 1,
  "time": 73178093,
  "ybase": 2010,
  "arg_id": "MyID",
  "arg_m": "stat",
  "arg_utc": -14400,
  "power": 61444.1,
  "emul": 2.608076793215e-03,
  "energy": {
    "now": [73178093, 3591206610],
    "hour": [[73177200, 3585450379], [73173600, 3562156429], [73170000, 3538603653]],
    "day": [[73108800, 3134920354], [73022400, 2563570839], [72936000, 1992302857],
    [72849600, 1419871564], [72763200, 877446316], [72676800, 305207065], [72590400, 211577],
    [72504000, 211577]],
    "days7": [[73108800, 3134920354], [72504000, 211577], [71899200, 211577]]
  }
}
```

Table 12: Data Structure Returned by the Realtime Statistics Query (sdata.json?m="stat")

Mnemonic	Type	Description
cmd	STR	Indicates the sdata.json query
arg_s	U8	Value set by the “s” parameter in the query. This will be the number of decimal places of accuracy for power values.
time	U32	The current sensor time, as of the time the query was being transmitted.
ybase	U16	Year base: See Section 5.2.
arg_id	STR	Value set by the “id” parameter in the query
arg_m	STR	Mode of the sdata.json query: “f1t”
arg_utc	U32	The UTC offset specified in the query.
power	U32 or F32	The instantaneous active power demand at the time of the query (the real-time power).
pmul	F64	Power scale factor used to scale raw power data for when “s” is not specified for the query.
energy.now[0]	U32	Time stamp for the current time

Mnemonic	Type	Description
energy.now[1]	INT48	Energy for the current time
energy.hour	Array	Contains an array of hourly time/energy values for the past 3 hours
energy.day	Array	Contains an array of daily time/energy values for the past 8 calendar days starting with today (reverse chronological order)
energy.days7	Array	Contains an array of 7-day time/energy values: start of today (energy.days7[0]), 7 days before that (energy.days7[1]) and another 7 days before that (energy.days7[2]) (reverse chronological order)

Energy used so far today (Watt-hours) =

$$\text{pmul} * (\text{energy.now}[1] - \text{energy.day}[0][1])$$

Energy used yesterday (Watt-hours) =

$$\text{pmul} * (\text{energy.day}[0][1] - \text{energy.day}[1][1])$$

Energy used the day before yesterday (Watt-hours) =

$$\text{pmul} * (\text{energy.day}[1][1] - \text{energy.day}[2][1])$$

Energy used for the past 7 days (Watt-hours) =

$$\text{pmul} * (\text{energy.days7}[0][1] - \text{energy.days7}[1][1])$$

Energy used for the previous 7 day period (Watt-hours) =

$$\text{pmul} * (\text{energy.days7}[1][1] - \text{energy.days7}[2][1])$$

With this information, the daily and weekly trends can be determined.

6.2.3 Datalog Data Query: datalog.json

The contents of the Datalog can be obtained through the query "datalog.json" and its alias datalog.csv.

The datalog works by storing time stamped sets of data in non-volatile memory at a set interval. Up to 47 electrical parameters (plus time) may be stored at intervals ranging from 15 seconds up to an hour. The selection of electrical parameters to be stored is controlled through 2 non-volatile system parameters ("logmask0", "logmask1").

The datalog can be configured through the "Datalog Configuration" tab inside the "Setup" page (Figure 2). This will correctly set the values for "logmask0", "logmask1", "logenable", and "logperiod".

Presets

Selection presets:

Log Control

Log Interval: 1 Minute ▼

Log Enable: Auto-enable after power-up/reset

Instantaneous Measurements

Voltage:	V(A) <input checked="" type="checkbox"/>	V(B) <input type="checkbox"/>	V(C) <input type="checkbox"/>
Current:	I(A) <input checked="" type="checkbox"/>	I(B) <input type="checkbox"/>	I(C) <input type="checkbox"/> I(D) <input type="checkbox"/>
Power:	P(A) <input type="checkbox"/>	P(B) <input type="checkbox"/>	P(C) <input type="checkbox"/> P(A+B+C) <input checked="" type="checkbox"/>
Volt-Amperes(VA):	VA(A) <input type="checkbox"/>	VA(B) <input type="checkbox"/>	VA(C) <input type="checkbox"/> VA(A+B+C) <input checked="" type="checkbox"/>
VA Reactive(VAR):	VAR(A) <input type="checkbox"/>	VAR(B) <input type="checkbox"/>	VAR(C) <input type="checkbox"/> VAR(A+B+C) <input checked="" type="checkbox"/>
Phase Angle:	Φ(A) <input type="checkbox"/>	Φ(B) <input type="checkbox"/>	Φ(C) <input type="checkbox"/>
Power Factor:	PF(A) <input type="checkbox"/>	PF(B) <input type="checkbox"/>	PF(C) <input type="checkbox"/> PF(A+B+C) <input checked="" type="checkbox"/>
Line Period:	T(A) <input type="checkbox"/>		

Cumulative Measurements (Total)

Active Energy:	WHr(A) <input type="checkbox"/>	WHr(B) <input type="checkbox"/>	WHr(C) <input type="checkbox"/>	WHr(A+B+C) <input checked="" type="checkbox"/>
Reactive Energy:	VARHr(A) <input type="checkbox"/>	VARHr(B) <input type="checkbox"/>	VARHr(C) <input type="checkbox"/>	VARHr(A+B+C) <input type="checkbox"/>
Apparent Energy:	VAHr(A) <input type="checkbox"/>	VAHr(B) <input type="checkbox"/>	VAHr(C) <input type="checkbox"/>	VAHr(A+B+C) <input type="checkbox"/>

Cumulative Measurements (Fundamental)

Active Energy:	WHr(A) <input type="checkbox"/>	WHr(B) <input type="checkbox"/>	WHr(C) <input type="checkbox"/>	WHr(A+B+C) <input type="checkbox"/>
Reactive Energy:	VARHr(A) <input type="checkbox"/>	VARHr(B) <input type="checkbox"/>	VARHr(C) <input type="checkbox"/>	VARHr(A+B+C) <input type="checkbox"/>

Changes Pending!:

Summary

Duration of Log: Each log entry requires 27 bytes of data storage. With a storage interval of 1 Minute, the total log duration is about **13.16** days.

Figure 2: Datalog Setup Form

Table 13: Datalog Query

Command	datalog.json or datalog.csv	
Parameters	id (STR)	This is an optional string that will be sent back with the response. It can be used by a requester to tag the response with an identifier. It has no other effect on the query.
	s (U8)	This is an optional integer parameter to specify that non-cumulative data should be scaled to standard units. The number specified sets the number of decimal places of precision. When the "s" parameter is not specified, all data is sent as raw data that must be scaled according to the scale factors available from the query.
	time (U32)	This optional parameter specifies a start time. Only data newer than "time" will be returned by the query. This parameter is used to capture only the latest data from the datalog. After the initial query, where no "time" parameter is specified (or a zero value), an application can make subsequent queries where "time" is set to the most recent time from the past query (the time at the end of the array). This prevents previous data from being transmitted, as only the most recent data will be sent.
	format	Determines the data format of the return data for the query. The default format is JSON. The alternate format of "csv" may be specified to have the query response transmitted as the excel compatible Comma Separated Vales (CSV) format.

6.2.3.1 Complete Datalog Data Query

When no "time" parameter is specified, a complete dump of all data available in the datalog is performed.

Example Query for datalog.json:

Query:

`http://netmeter/datalog.json?s=1&i d=zzzzz`

Example Response when the datalog is disabled ("logenable":0) and empty ("logdata" has no data):

```
{
  "cmd": "datalog",
  "arg_id": "zzzzz",
  "arg_s": 1,
  "model": "NetMeter-3P-600-100",
  "mac": "00:04:A3:50:3F:94",
  "label": "NetMeter98",
  "desc": "NetMeter Unit Number 98",
  "time": 72861376,
  "ybase": 2010,
  "logenable": 0,
  "logperiod": 15,
  "logmask0": "0x11111111",
  "logmask1": "0x1",
  "names": ["time", "V(A)", "I(A)", "P(A)", "VA(A)", "VAR(A)", "PHI(A)", "PF(A)", "T(A)", "WHR(A)"],
  "units": ["s", "Volts RMS", "Amps RMS", "Watts", "Volt-amps", "Watts", "Degrees", "Percent", "us", "Watt-Hours"],
  "scale": [1, 1, 1, 1, 1, 1, 1, 1, 1, 2.608076793215e-03],
  "logdata": []
}
```

Example Response when the datalog is enabled and has some data to report:

```
{
  "cmd": "datalog",
  "arg_id": "zzzzz",
  "arg_s": 1,
  "model": "NetMeter-3P-600-100",
  "mac": "00:04:A3:50:3F:94",
  "label": "NetMeter98",
  "desc": "NetMeter Unit Number 98",
  "time": 72866536,
  "ybase": 2010,
  "logenable": 1,
  "logperiod": 15,
  "logmask0": "0x11111111",
  "logmask1": "0x1",
  "wrptr": "0x4022E",
  "rdptr": "0x40000",
  "names": ["time", "V(A)", "I(A)", "P(A)", "VA(A)", "VAR(A)", "PHI(A)", "PF(A)", "T(A)", "WHR(A)"],
  "units": ["s", "Volts RMS", "Amps RMS", "Watts", "Volt-amps", "Watts", "Degrees", "Percent", "us", "Watt-Hours"],
  "scale": [1, 1, 1, 1, 1, 1, 1, 1, 1, 2.608076793215e-03],
  "logdata": [
    [72866415, 224.8, 93.4, 20991.1, 20989.4, 3.0, -0.1, 100.0, 16664.1, 506702873],
    [72866430, 224.0, 93.0, 20840.7, 20840.0, 4.9, 0.0, 100.0, 16664.1, 506736362],
    [72866445, 224.0, 93.0, 20839.9, 20840.6, 4.2, 0.0, 100.0, 16660.2, 506769712],
    [72866460, 224.5, 93.2, 20933.1, 20934.8, 3.0, 0.0, 100.0, 16664.1, 506803211],
    [72866475, 224.2, 93.1, 20881.3, 20882.5, 4.6, 0.0, 100.0, 16664.1, 506836746],
    [72866490, 225.0, 93.5, 21029.3, 21031.1, 3.8, -0.1, 100.0, 16668.0, 506870234],
    [72866505, 224.9, 93.4, 21005.2, 21007.5, 5.4, -0.1, 100.0, 16668.0, 506903797],
    [72866520, 225.0, 93.5, 21033.1, 21032.7, 4.4, 0.0, 100.0, 16668.0, 506937348],
    [72866535, 224.7, 93.3, 20978.3, 20976.5, 5.3, -0.1, 100.0, 16668.0, 506970847]
  ]
}
```

The JSON data members are described in Table 14:

Table 14: Description of the Data Structure Returned by the "datalog.json" Query

Mnemonic	Type	Description
cmd	STR	Specifies the datalog command
arg_id	STR	Value set by the "id" parameter in the query
arg_s	U8	Value set by the "s" parameter in the query
model	STR	Hardware model number of the Z3 Controls sensor

Mnemonic	Type	Description
mac	MAC	Gateway MAC address. Helps to identify where the data came from.
label	STR	The user defined label for the NetMeter sensor
desc	STR	The user defined description of the NetMeter sensor
time	U32	The current sensor time as of the time the query was transmitted.
ybase	U16	Year base: See Section 5.2.
logenable	BOOL	Datalog enable: 0 when the data logger is disabled, 1 when it is enabled. It is enabled/disabled through the sinfo.json query. It is strongly recommended that you use the built-in web setup in order to set this up properly.
logperiod	U16	Log interval (in seconds). The log period is set through the sinfo.json query. It is strongly recommended that you use the built-in web setup in order to set this up properly.
logmask0	U32	A bitmask of what electrical parameters are to be recorded by the data logger (part 1). It is strongly recommended that you use the built-in web setup in order to set this up properly.
logmask1	U32	A bitmask of what electrical parameters are to be recorded by the data logger (part 2). It is strongly recommended that you use the built-in web setup in order to set this up properly.
names	STR	An array containing the names of the data items in the datalog.
units	STR	An array containing the units of the data items in the datalog.
scale	F64	An array containing the scale factor that must be applied to each data member in order to convert it into the units specified in "units"
logdata	Mixed	An array containing time sequenced arrays of datalog data. Each member of the inner arrays corresponds to the set of "names", "units", and "scale".

When the "csv" format is used, the same data as above is formatted as follows:

Query:

<http://netmeter/datalog.json?s=1&format=csv>

Example Response:

```
arg_s, 1
model, NetMeter- 3P- 600- 100
mac, 00: 04: A3: 50: 3F: 94
label, NetMeter98
desc, NetMeter Unit Number 98
time, 72866684
ybase, 2010
logenable, 1
logperiod, 15
logmask0, 0x11111111
logmask1, 0x1
wrptr, 0x4022E
rdptr, 0x40000
time, V(A), I(A), P(A), VA(A), VAR(A), PHI(A), PF(A), T(A), WHr(A)
s, Volts RMS, Amps RMS, Watts, Volt-amps, Watts, Degrees, Percent, us, Watt-Hours
1, 1, 1, 1, 1, 1, 1, 1, 1, 2. 608076793215e- 03, scale
72866415, 224. 8, 93. 4, 20991. 1, 20989. 4, 3. 0, - 0. 1, 100. 0, 16664. 1, 506702873
72866430, 224. 0, 93. 0, 20840. 7, 20840. 0, 4. 9, 0. 0, 100. 0, 16664. 1, 506736362
72866445, 224. 0, 93. 0, 20839. 9, 20840. 6, 4. 2, 0. 0, 100. 0, 16660. 2, 506769712
72866460, 224. 5, 93. 2, 20933. 1, 20934. 8, 3. 0, 0. 0, 100. 0, 16664. 1, 506803211
72866475, 224. 2, 93. 1, 20881. 3, 20882. 5, 4. 6, 0. 0, 100. 0, 16664. 1, 506836746
72866490, 225. 0, 93. 5, 21029. 3, 21031. 1, 3. 8, - 0. 1, 100. 0, 16668. 0, 506870234
72866505, 224. 9, 93. 4, 21005. 2, 21007. 5, 5. 4, - 0. 1, 100. 0, 16668. 0, 506903797
72866520, 225. 0, 93. 5, 21033. 1, 21032. 7, 4. 4, 0. 0, 100. 0, 16668. 0, 506937348
72866535, 224. 7, 93. 3, 20978. 3, 20976. 5, 5. 3, - 0. 1, 100. 0, 16668. 0, 506970847
```

6.2.3.2 Incremental Datalog Data Query

In order to minimize data transfer, incremental log data may be gathered after an initial full data query is performed as described in the previous section.

Incremental queries are performed by specifying the “time” parameter in the command:

Incremental Query (JSON mode):

```
http://netmeter/datalog.json?s=1&time=72866535&id=zzzzz
```

Example Response:

```
{
  "cmd": "datalog",
  "arg_id": "zzzzz",
  "arg_s": 1,
  "logdata": [
    [72866550, 225. 0, 93. 5, 21036. 7, 21034. 9, 4. 5, 0. 0, 100. 0, 16664. 1, 507004418],
    [72866565, 224. 5, 93. 2, 20928. 2, 20928. 4, 5. 5, 0. 0, 100. 0, 16660. 2, 507037950]
  ]
}
```

In the case above, an additional 2 time samples of data have been added to the datalog since time = 72866535.

The exact same result for the CSV format is:

Incremental Query (CSV mode):

```
http://netmeter/datalog.json?s=1&time=72866535&format=csv
```

Example Response:

```
72866550, 225. 0, 93. 5, 21036. 7, 21034. 9, 4. 5, 0. 0, 100. 0, 16664. 1, 507004418
72866565, 224. 5, 93. 2, 20928. 2, 20928. 4, 5. 5, 0. 0, 100. 0, 16660. 2, 507037950
```

This response can simply be appended to the previous CSV response in order to have a complete record.



www.z3controls.com

support@z3controls.com

Phone: 1-877-454-4436

4261-A14 Highway #7 East, Unit 290
Markham, ON L3R 9W6
Canada

Copyright © 2011 Z3 Controls Inc.