



# Virtual Smart-Net Gateway

Software User's Manual

## Table of Contents

<b>Overview .....</b>	<b>5</b>
Features .....	5
Contact Information .....	5
<b>Installation .....</b>	<b>6</b>
Installing Java .....	6
Windows .....	6
Linux - RPM .....	7
Installing the Smart-Net Port Adapter .....	8
Installing RXTX on Linux .....	8
Connecting Smart-Net Devices.....	8
Installing the Virtual Smart-Net Gateway Software.....	10
Windows .....	10
Linux .....	10
<b>Configuration .....</b>	<b>11</b>
Overview.....	11
General Options .....	11
Configuration File.....	11
Smart-Net Adapter Type.....	12
Adapter Port .....	12
Apply.....	12
Time Options .....	12
Setting the Time Zone.....	13
Using the System Time .....	13
Using a Manually Entered Time .....	13
Using an Automatically Obtained Time .....	14
Device History .....	14
Alerting Options.....	15
Enabling/Disabling .....	16
SMTP Mail Server .....	16
From.....	16
To .....	16

Alert Type .....	16
Log Collection Options .....	16
Enabling/Disabling .....	17
Log Collection Interval .....	17
Log File Directory .....	17
Log Filename Prefix .....	17
View Logs .....	18
Erase Logs .....	18
FTP Log Upload Options .....	18
Enabling/Disabling .....	19
Server & Port .....	19
Username & Password .....	19
Remote Directory .....	20
FTP Upload Interval .....	20
Testing .....	20
XML-RPC Server Options .....	20
Enabling/Disabling .....	20
Server Port .....	21
XML-RPC Server Path .....	21
<b>XML-RPC .....</b>	<b>21</b>
Overview .....	21
Accessing XML-RPC Methods .....	22
Common Field Descriptions .....	22
Timestamp .....	22
Status .....	22
ROMID .....	23
ParentID .....	23
UserID .....	23
Smart-Net Gateway XML-RPC Methods .....	24
SmartNet.Status .....	24
SmartNet.Time .....	24
SmartNet.Find .....	24
SmartSenseTH.Read .....	25

SmartSenseTH.Write .....	26
SmartWatt.Read .....	27
SmartWatt.Write .....	30
SmartPDU.Read.....	31
SmartPDU.Map.....	32
SmartPDU.Write.....	32
Additional Resources .....	33
<b>Log Collection.....</b>	<b>34</b>
Overview.....	34
Log File Formats.....	34
Common Fields.....	34
SmartSenseTH .....	35
SmartWatt .....	36
SmartPDU .....	38
Smart-Net.....	39
Exception .....	39
Test.....	39
Intervals.....	40

## Overview

### *Features*

The Virtual Smart-Net Gateway software offers the following features:

- **Platform Independent** – The Virtual Smart-Net Gateway software was written in Java, meaning it has the ability to run on any platform where Java is supported.
- **Plug and Play** – The Virtual Smart-Net Gateway software will automatically discover newly added Smart-Net Devices.
- **Simple Interface** – The Virtual Smart-Net Gateway software provides an easy to use interface for configuration.
- **Automatic Logging** – The Virtual Smart-Net Gateway software can automatically monitor and log Smart-Net Device information at specified intervals in comma-delimited text files.
- **FTP Uploading** – The Virtual Smart-Net Gateway software can automatically upload collected log files at a specified interval to a remote FTP server.
- **XML-RPC** – The Virtual Smart-Net Gateway software provides an XML-RPC server for remote access to the Smart-Net Device Network.

### *Contact Information*

If you wish to contact Smart Works, Inc., you may do so using one of the following methods:

- **E-Mail** – [info@smart-works.com](mailto:info@smart-works.com)
- **Telephone** – (253) 735-0552
- **Fax** – (253) 735-0562
- **Web** – <http://www.smart-works.com>

## Installation

In order to use the Virtual Smart-Net Gateway, your system must be prepared in order to meet the software requirements of the program. This includes:

- Installing the Java Runtime Environment (JRE) or Java Development Kit (JDK) for your platform (page 6),
- Installing the Smart-Net Port Adapter drivers and/or Java classes (page 8),
- Connecting your Smart-Net Devices (page 8),
- Installing the Virtual Smart-Net Gateway program (page 8), and
- Configuring the Virtual Smart-Net Gateway program (page 11).

### Installing Java

The Virtual Smart-Net Gateway software was written for Java. Therefore, it is required that your system has the Java Runtime Environment installed to use this software.

### Windows

To install the Java Runtime Environment for Windows, follow these steps:

1. Open the directory “3rd Party\JRE\Windows” in Windows Explorer.
2. Double-click “jre-1\_5\_0\_01-windows-i586-p.exe” to begin the Java Runtime Environment (JRE) installation.
3. Read and accept the license agreement and click **Next**.
4. Select **Typical** and click **Next** to begin the installation (Figure 1).

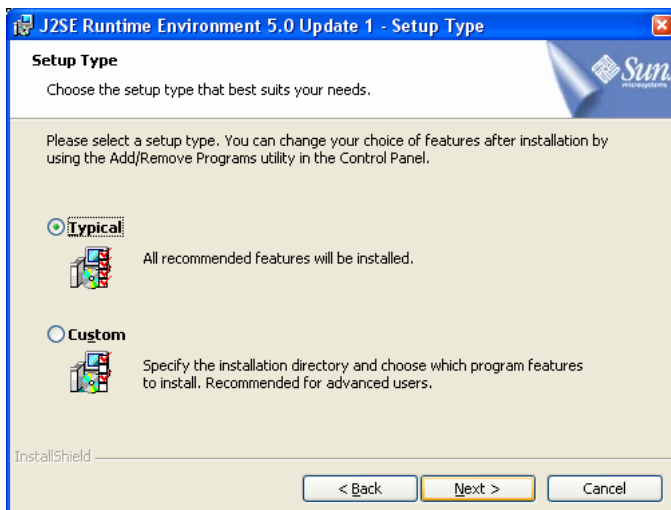
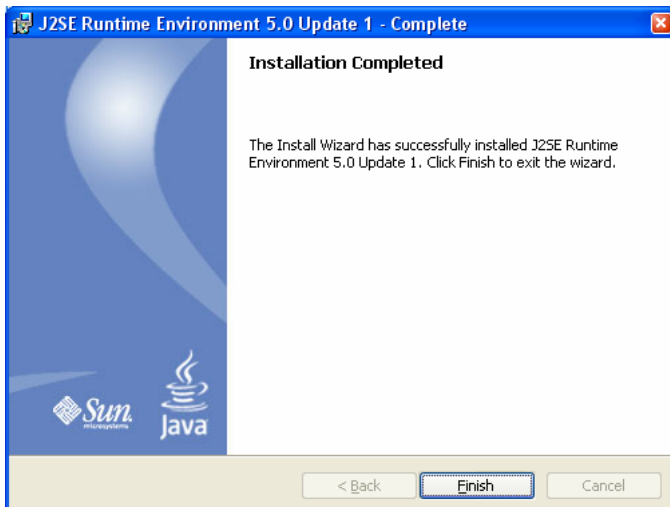


Figure 1. Setup Type

5. When the installation is completed click **Finish** (Figure 2).



**Figure 2.** Installation Completed

6. Proceed to the next section, “Installing the Smart-Net Port Adapter”.

## Linux - RPM

**Note:** The installation of RXTX requires the Java Software Development Kit (JDK). Therefore, these instructions show the installation of the JDK (which also includes the JRE). To install the JDK for Linux, follow these steps:

1. Log in as **root**.
2. Mount the CD-ROM:  
# mount /dev/cdrom /mnt/cdrom
3. Change to a temporary directory. For example:  
# cd /tmp
4. Copy the file “Virtual Smart-Net Gateway/3rd Party/JDK/Linux/jdk-1\_5\_0\_01-linux-i586-rpm.bin” on the CD-ROM to the temporary directory. For example:  
# cp /mnt/cdrom/Virtual\ Smart-Net\ Gateway/3rd\ Party/JDK/Linux/jdk-1\_5\_0\_01-linux-i586-rpm.bin /tmp/
5. Ensure the file you just copied is executable. For example:  
# chmod a+x /tmp/jdk-1\_5\_0\_01-linux-i586-rpm.bin
6. Execute the file “jdk-1\_5\_0\_01-linux-i586-rpm.bin”. For example:  
# ./jdk-1\_5\_0\_01-linux-i586-rpm.bin
7. Read and agree to the license agreement.
8. Wait for the installation to complete.
9. Add the directory “/usr/java/jdk1.5.0\_01/bin” to your PATH environment variable (i.e. by editing the file /etc/profile). **Note:** You may need to log out and back in for this change to take effect.
10. Proceed to the next section, “Installing the Smart-Net Port Adapter”.

## ***Installing the Smart-Net Port Adapter***

To use the Virtual Smart-Net Gateway software, the appropriate drivers must be installed for the USB or Serial Port Smart-Net Adapter. To install the correct drivers, determine your operating system (i.e. Windows or Linux) and the type of Smart-Net Adapter you have (USB or Serial) and follow the appropriate instructions.

If you have...

- **Windows and a USB or Serial Smart-Net Adapter** then the drivers for the Smart-Net Port adapter will be automatically installed with the install wizard for the Virtual Smart-Net Gateway. Do not plug in the adapter until this software is installed.
- **Linux and a USB Smart-Net Adapter** – This adapter is not yet supported under Linux.
- **Linux and a Serial Smart-Net Adapter** then you must follow the instructions below for “Installing RXTX on Linux”.
- **Other** – If your platform supports RXTX for Java, you may be able to use the Serial Smart-Net Adapter.

## **Installing RXTX on Linux**

1. Mount the CD-ROM.
2. Extract the file “3rd Party/RXTX/Linux/rxtx-devel.tar.gz”. For example:  
\$ tar zxvf 3rd\ Party/RXTX/Linux/rxtx-devel.tar.gz
3. Change to the directory the files were extracted to. For example:  
\$ cd rxtx-devel
4. Type the following commands:  
\$ ./configure  
\$ make
5. Become root:  
\$ su
6. Type the following command:  
# make install
7. Proceed to the next section, “Connecting Smart-Net Devices”.

## ***Connecting Smart-Net Devices***

The Smart-Net Gateway has a built-in RJ-11 connector that controls a Smart-Net Device Network. Every Smart-Net device has two RJ-11 connectors to allow for daisy chaining of devices. Simply connect one of the devices on the Smart-Net Network to the RJ-11 port on the Smart-Net Gateway. The devices will automatically be detected and no additional configuration is necessary.

There are a few points to keep in mind when installing a Smart-Net Network:

- **Polarity** – The Smart-Net network uses standard RJ-11 connectors to make it easy and convenient to wire and extend. If standard telephone cable is used, it is important to ensure that it is not of a “flipped” type. The cable must be “straight-through”, that is pin 1 on one end of the cable must conduct directly to pin 1 on the other end of the cable. The same goes for all the pins on the cable.
- **Topology** – The recommended topology of wiring devices in the Smart-Net Network is in a “daisy chain” fashion. That is, each additional device should extend from the last device on the network. Wiring the network this way will ensure reliable communication and the longest distances. Branching off to other groups of Smart-Net devices from one point will cause signal errors and greatly deteriorate the reliability and maximum distance of the network.
- **Cable Grade** – Using Category 5 or Category 6 wire will support the longest distances with minimal communication errors. However, if the network is very short, lower grade cable may be used.

## ***Installing the Virtual Smart-Net Gateway Software***

To run the Virtual Smart-Net Gateway software, it must be copied to the hard drive of the system and executed with the Java Runtime Environment. Instructions to do this on Windows and Linux follow.

### **Windows**

1. Open the directory “Virtual Smart-Net Gateway” on the CD-ROM with Windows Explorer.
2. Open the directory “Windows Installer” subdirectory.
3. Execute the file “Setup.exe”.
4. Follow the instructions in the Virtual Smart-Net Gateway Setup Wizard. The wizard will copy the files to your hard drive, install the Smart-Net Port Adapter drivers, and create a start-menu shortcut for the user’s manual and for the program.

### **Linux**

**Note:** The user that will run the Virtual Smart-Net Gateway software must have permissions to write to its configuration file (by default smartnet.conf) and the log directory (by default logs/). Additionally, it must have access to the serial port by making the user a member of the “uucp” and “lock” groups.

1. Start X Windows and open a shell/xterm.
2. Change to the directory “Virtual Smart-Net Gateway” on the CD-ROM.
3. Copy the all the files in this directory to a permanent location, such as “/usr/local/smartnetgw”.
4. Change to the directory you just copied the files to.
5. Execute the jar file with the java command:  
\$ java -jar SmartNetGateway.jar  
- or -  
\$ java -cp ./SmartNetGateway.jar SmartNetGateway

## Configuration

### Overview

When the Virtual Smart-Net Gateway is started for the first time, it will automatically create a subdirectory “Logs” in the program directory and a file “smartnet.conf”. By default, Log Collection, FTP Log Uploading, and the XML-RPC Server are all disabled. Additionally, it will default to a Serial Smart-Net Port adapter on the system’s first serial port.

### General Options

When the Virtual Smart-Net Gateway’s Graphical User Interface (GUI) is loaded, it will automatically show the “General Options” pane (Figure 3). Changes can be made as described below. When these options are configured, the **Apply** button must be clicked to save and activate the changes.

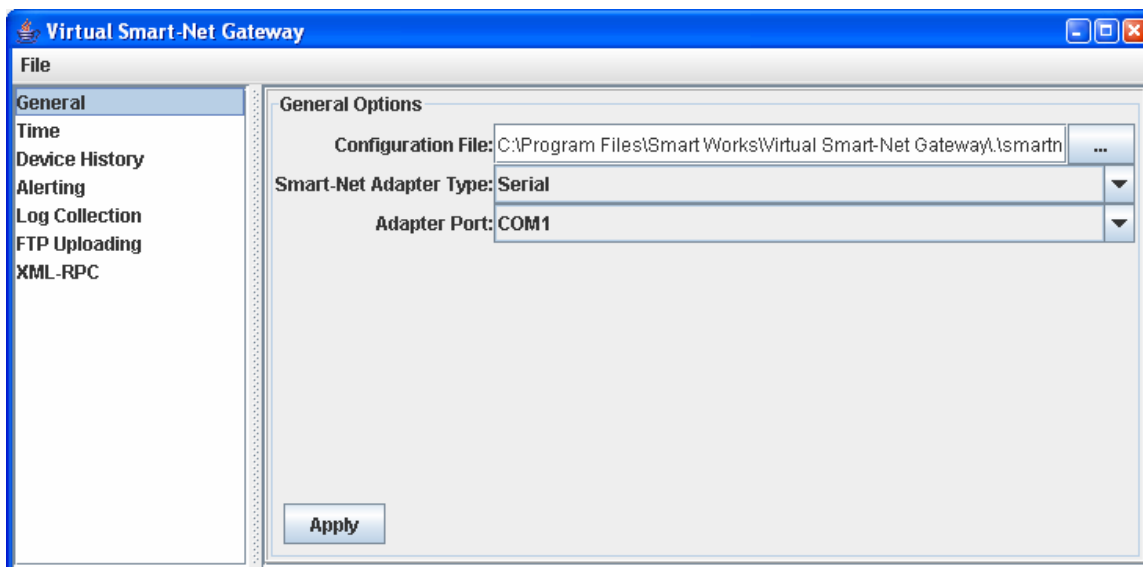


Figure 3. General Options

### Configuration File

By default, the Virtual Smart-Net Gateway reads its configuration from the file “smartnet.conf” in the same directory that it was started from. If you wish to load another configuration file for this session, type the path and name of the file in this field or click the “...” button to browse for a configuration file and click **Apply**.

Note that when the **Apply** button is clicked, any other unsaved changes will be lost and new settings will be loaded from the file specified. If the file does not exist or contains invalid configuration information, it will be created and overwritten with new default settings.

## Smart-Net Adapter Type

If you have a Serial Smart-Net Port Adapter plugged into a Serial port on your system, select “Serial”. If you have the USB Smart-Net Port Adapter, plug it in to an available USB port and select “USB”. If there is no option for USB then your platform does not support a USB Smart-Net Port Adapter or the drivers are installed incorrectly.

## Adapter Port

If you selected “Serial” for the Smart-Net Adapter Type, choose the proper serial port from this list. If there are no serial ports listed under Windows, check that the drivers were installed properly via the installation wizard. If there are no serial ports listed under Linux, check that RXTX was properly installed and the current user has correct permissions to use the serial ports (members of group “uucp” and “lock”).

If you selected “USB” for the Smart-Net Adapter Type, this option will be “Default”. The first detected USB Smart-Net Adapter in the system will be used.

## Apply

Any changes in the General Options must be applied or your settings will be lost. The Apply button will also attempt to detect the selected Smart-Net Port Adapter. If the adapter is not found, a message box will be displayed (Figure 4).

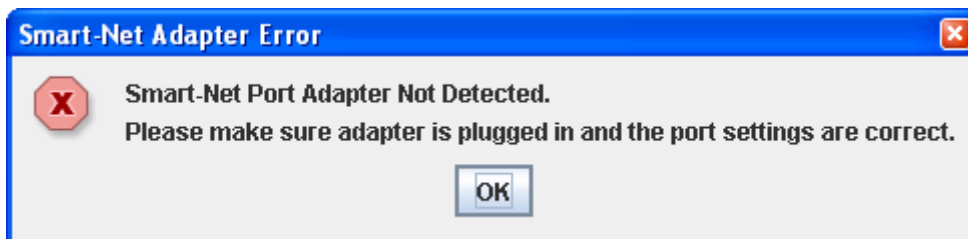
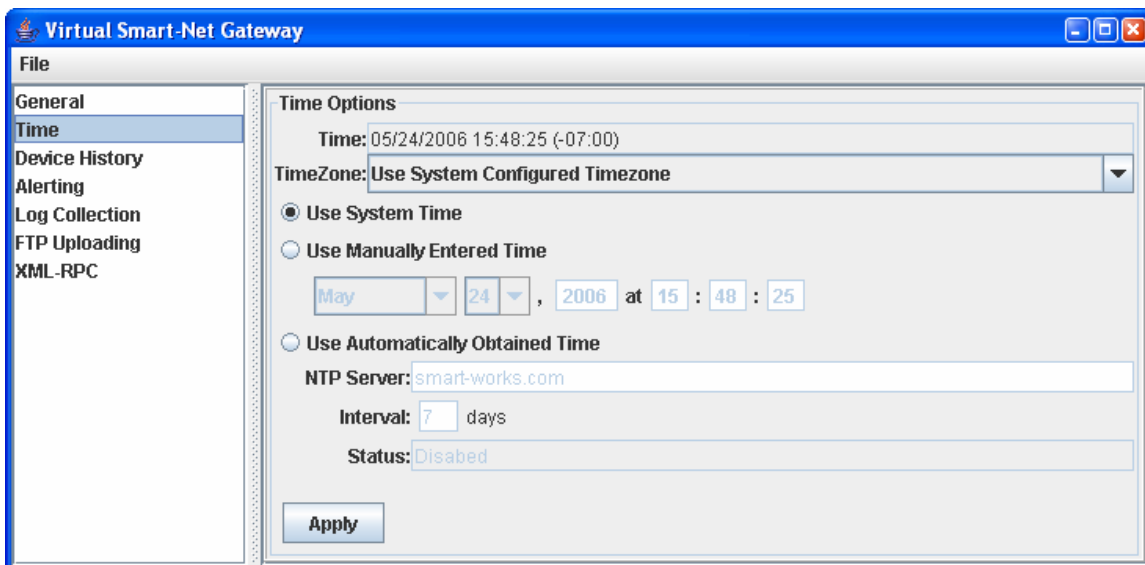


Figure 4. Smart Net Port Adapter Not Detected

If you see this message, ensure the correct adapter type and port are selected, that the device is plugged in, and that the Smart-Net network is wired correctly.

## Time Options

Click “Time” on the left column to open the “Time Options” pane (Figure 5). Changes to this feature of the Virtual Smart-Net Gateway can be made as described below. When these options are configured, the **Apply** button must be clicked to save and activate the changes.



**Figure 5.** Time Options

The time options allow for changing the source of the timestamps used in the log files to either a manually entered time or a network time server. This allows the Virtual Smart-Net Gateway’s log file to be synchronized with that of other Smart-Net Gateways or servers without affecting the system time clock of the server it is running on.

Additionally, the time options allow for changing the time zone that the log files are recorded in.

## Setting the Time Zone

The “TimeZone” drop down box displays all the possible time zone settings for the Virtual Smart-Net Gateway. By default, it is set to “Use System Configured Timezone”, which is the first entry in the drop down box. The default option obtains the appropriate time zone information from your operating system settings. If the operating system’s time zone setting is incorrect, or if the log files should be time stamped in a remote time zone different from the operating system, then select the appropriate time zone from the list. Click **Apply** to activate the new time zone settings.

## Using the System Time

The “Use System Time” option instructs the Virtual Smart-Net Gateway to write all time stamps in the log files using the system clock. When this option is set, the system clock should be set correctly for accurate log readings.

## Using a Manually Entered Time

The “Use Manually Entered Time” option allows the Virtual Smart-Net Gateway to use a time different from the system clock. This option can be used if the system clock is not properly set or if you need to ensure that multiple Smart-Net Gateways all log using the same relative time.

When this option is selected, the date and time can be entered using the fields for the month, day, year, hour, minute, and second. The year should be entered using 4 digits

(i.e., “2006”, not “06”). The time entered is in 24-hour format, where midnight is “00” and 11 PM is “23”. Once the correct time is entered, click the **Apply** button to set the new time.

Note that if this option is selected and the host system clock is changed, the manually entered time setting will need to be adjusted again.

## Using an Automatically Obtained Time

The “Use Automatically Obtained Time” option instructs the Virtual Smart-Net Gateway to use a remote network time server’s time instead of the system clock. This option can be used to automatically synchronize multiple Smart-Net Gateways to a single time source without affecting the system clock.

When this option is selected, the following options are configurable:

- **NTP Server** – The hostname or IP Address of the network time protocol server to synchronize to. By default, this is set to “smart-works.com”.
- **Interval** – The number of days the Virtual Smart-Net Gateway waits from the previous synchronization before synchronizing again with the NTP Server. By default, this is set to 7 days (1 week).

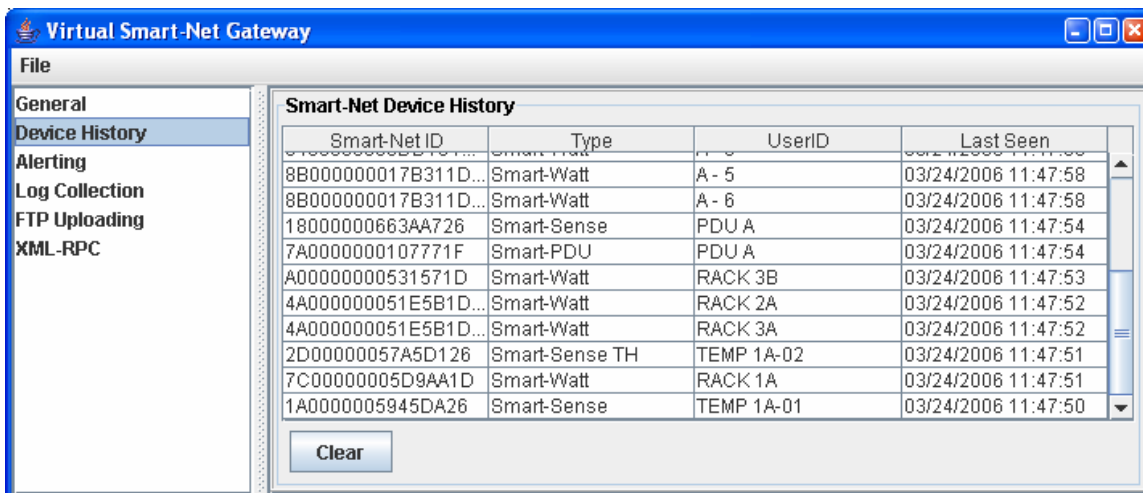
Additionally, when this option is selected, the status of the last synchronization will be visible. If the last synchronization was successful, it will display the time the synchronization occurred. Otherwise, if the last synchronization attempt failed, it will display an appropriate message indicating the failure.

When these options are changed, click the **Apply** button to synchronize the time according to the new settings.

Note that if this option is selected and the host system clock is changed, the time must be resynchronized in order to maintain its accuracy.

## Device History

To view a table of the history of connected devices, click “Device History” on the left column to open the “Smart-Net Device History” pane (Figure 6). The Smart-Net Device History shows all the devices the Log Collection and Alerting function has seen since the last time the program was started or the list was cleared. The table provides a convenient way to view the Smart-Net ID, Smart-Net Device Type, UserID (if applicable), and the Date/Time the device was last seen on the network. To sort the table, simply click on the desired heading label.

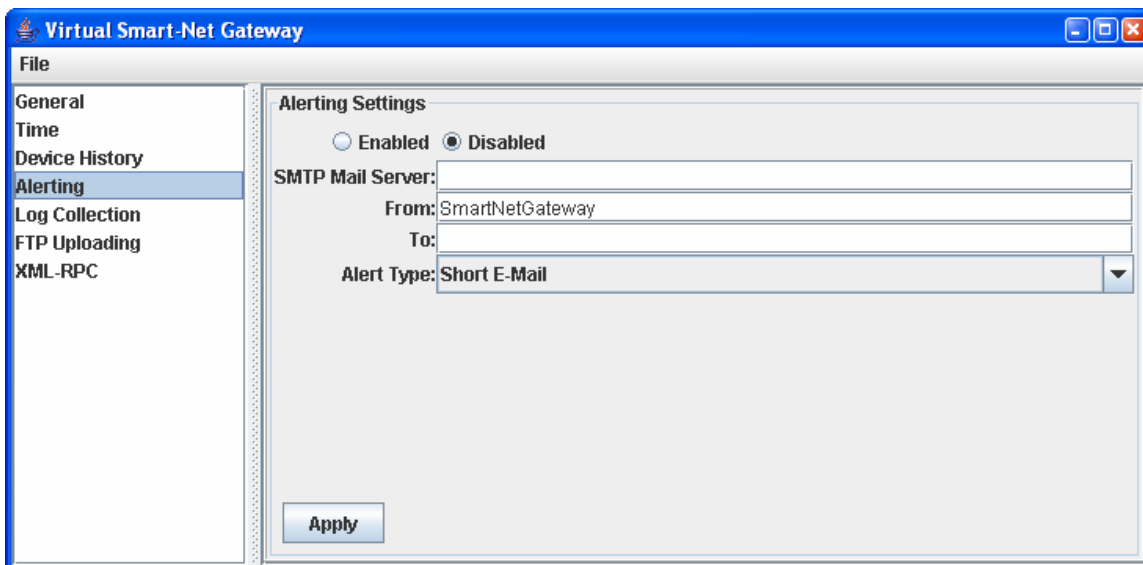


**Figure 6.** Smart-Net Device History

To clear the Smart-Net Device History, click the Clear button below the table. The Log Collection function or Alerting function must be enabled for this table to be updated.

## Alerting Options

Click “Alerting” on the left column to open the “Alerting Options” pane (Figure 7). Changes to the Alerting feature of the Virtual Smart-Net Gateway can be made as described below. When these options are configured, the **Apply** button must be clicked to save and activate the changes.



**Figure 7.** Alerting Options

The Alerting feature will automatically send an E-Mail alert every time a Smart-Net device is considered to be in an “Alerting” or “Alarming” state. Each Smart-Net device has its own individual alert settings that are stored directly in the device. To configure an individual Smart-Net device’s settings, you may use the XML-RPC feature.

## Enabling/Disabling

To enable the ability to send alerts, choose “Enabled”. Otherwise, choose “Disabled” to disable alert sending. Note that if none of the Smart-Net devices are configured to allow alerts, no alerts will be sent even if this feature is “Enabled”. By default, this feature is disabled.

## SMTP Mail Server

The “SMTP Mail Server” field specifies the hostname or IP address of a Simple Mail Transport Protocol (SMTP) server in which to relay the alert E-Mails to. The specified SMTP server must allow this computer to send e-mails to the desired address. If the server does not allow relaying, then the alert message might be rejected by the server.

For example, the SMTP Mail Server to use might be “mail.example.net”.

## From

The “From” field specifies the E-Mail address that the alert messages should appear to have originated from. By default, this is “SmartNetGateway”. However, it may be desired to change it to an address to help identify the Virtual Smart-Net Gateway that the alert was sent from. For example, the “From” field may be set to:

VirtualGateway0001

VirtualGateway0001@example.net

Note that some mail servers might reject messages with an invalid “From” address causing alert messages to not be delivered.

## To

The “To” field specifies the E-Mail address to send the alert messages to. For example, alerts@example.net may be entered. Note that some mail servers disallow relaying to foreign E-Mail addresses causing alert messages not to be delivered.

## Alert Type

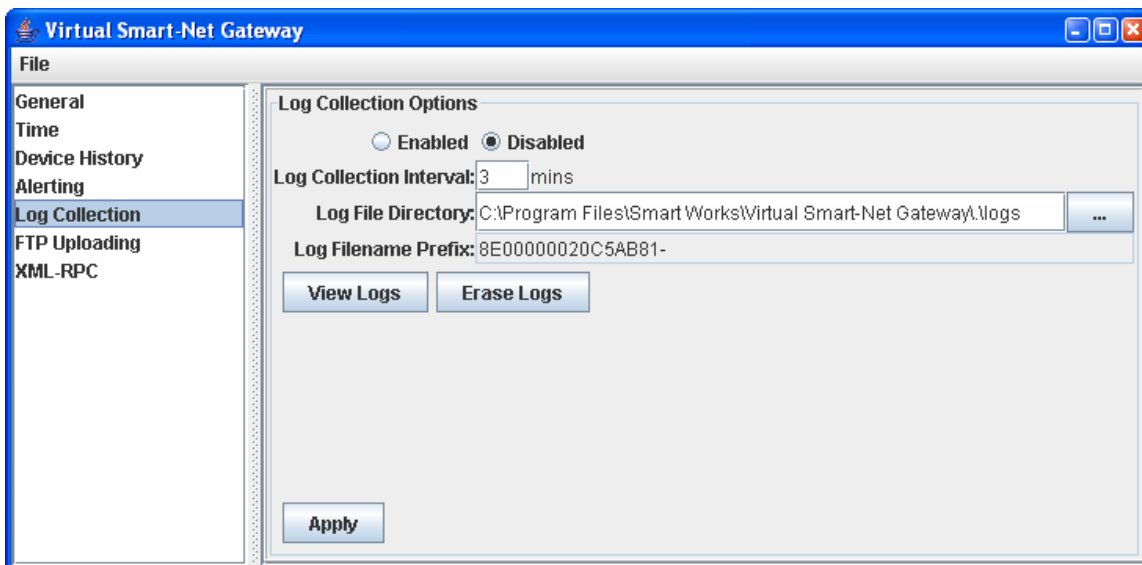
The “Alert Type” setting changes the type of E-Mail that is sent to the recipient.

Selecting “Short E-Mail” will send alert messages that are very brief and suitable for delivery to a cell phone or a pager.

Selecting “Normal E-Mail” will send alert messages that contain every detail about the device and the alert condition, suitable for reading in a standard e-mail client.

## Log Collection Options

Click “Log Collection” on the left column to open the “Log Collection Options” pane (Figure 8). Changes to the Log Collection feature of the Virtual Smart-Net Gateway can be made as described below. When these options are configured, the **Apply** button must be clicked to save and activate the changes.



**Figure 8.** Log Collection Options

The Log Collection feature will automatically discover all connected Smart-Net devices and log their status at the specified interval to a text file in the specified directory.

For more information on the log file format, refer to the section “Log File Formats” below.

## Enabling/Disabling

To enable automatic log collection, choose “Enabled”. Otherwise, choose “Disabled” to disable automatic log collection. By default, this is disabled.

## Log Collection Interval

Enter the number of minutes between reading Smart-Net Device information and logging. By default, this is 3 minutes.

## Log File Directory

By default, the Virtual Smart-Net Gateway uses the directory “logs” in the directory the program was started from to store its log files. To change this directory, type in its path or click the “...” button and browse for a directory. When the **Apply** button is clicked, all logging will take place in the new directory and any log files in the old directory will not be viewable or erasable with the View Logs and Erase Logs buttons.

Note that if a directory is selected that already contains files, they could be overwritten or deleted. Therefore, an empty directory should be chosen. If the selected directory does not exist, it will be created.

## Log Filename Prefix

The Log Filename Prefix is the first part of the log file names of each Smart-Net Device and is automatically generated from the serial number of the Smart-Net Port Adapter.

The prefix is used to uniquely identify a particular Virtual Smart-Net Gateway in case multiple gateways are uploading to the same file server.

For example, if the prefix is “CA0000002085C981-” then log files might be generated with the names:

- CA0000002085C981-SmartWatts.txt
- CA0000002085C981-SmartSenseTHs.txt
- CA0000002085C981-Exception.txt

## View Logs

To quickly view the contents of the current log files, click the **View Logs** button. A Window will appear with a drop down menu of the log files and the contents of the log file (Figure 9). When finished viewing the log files, close the window.

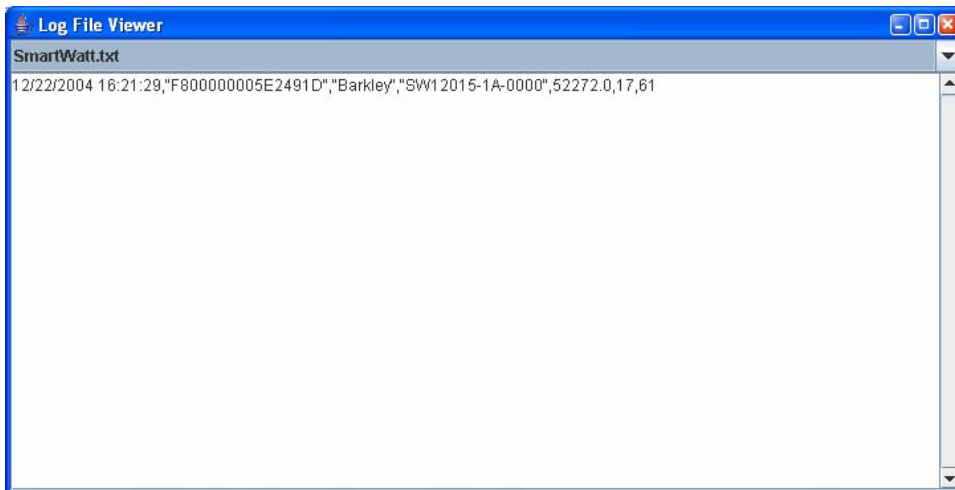


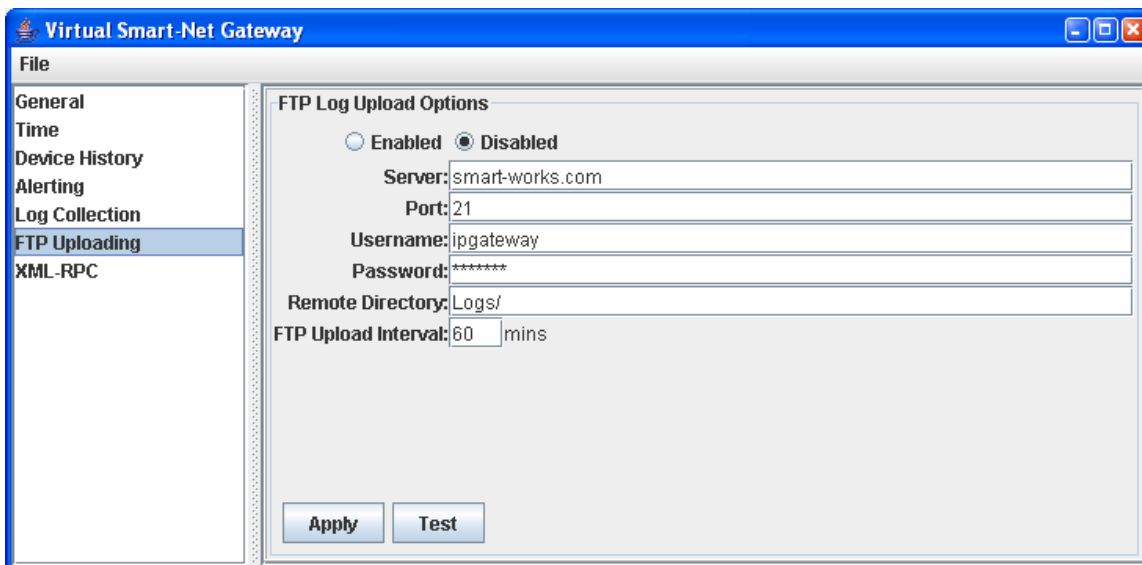
Figure 9. Log File Viewer

## Erase Logs

To erase the current log files, click the **Erase Logs** button. This will scan the Log File Directory for files that begin with the Log Filename and delete them. If log collection is enabled, new files will be created automatically on the next Log Collection Interval.

## FTP Log Upload Options

Click “FTP Uploading” in the left column to change options for FTP Log Uploading (Figure 10). When these options are configured, the **Apply** or **Test** button must be clicked for the changes to take effect.



**Figure 10.** FTP Log Upload Options

When FTP Log Upload is enabled, it will automatically scan the Log File Directory for log files, upload these files to a remote FTP server, and delete the log files locally.

### Enabling/Disabling

To enable log file uploading, choose “Enabled”. Otherwise, choose “Disabled” to disable log file uploading. This change does not take effect until **Apply** or **Test** is clicked.

By default, this option is disabled.

**Note:** When **Test** is clicked, the log files are immediately uploaded regardless of this option.

### Server & Port

The “Server” field specifies the hostname or IP address of the FTP Server in which the Smart-Net Gateway should upload its log data to. By default, it is set to “smart-works.com” for a public server.

The “Port” field specifies which port the remote FTP Server is listening on. By default, FTP uses port 21. This value should only be changed if your FTP Server uses a non-standard port number.

### Username & Password

If the FTP server that the Smart-Net Gateway connects to uses an anonymous login, then “anonymous” should be entered for the username and an e-mail address for the password. If the FTP server has restricted address, then the proper username/password should be filled in here. Note that FTP is an insecure protocol and the password will be transmitted in clear-text across the network on each upload attempt.

**Warning:** The password is stored in clear-text in the configuration file.

## Remote Directory

A directory on the FTP server to save files to may be specified here. By default, this is “Logs/” and may be case sensitive, depending on the operating system of the FTP Server.

## FTP Upload Interval

The upload interval is the number of minutes the Virtual Smart-Net Gateway will wait between log file upload attempts. By default, this is set to 60 minutes (1 hour).

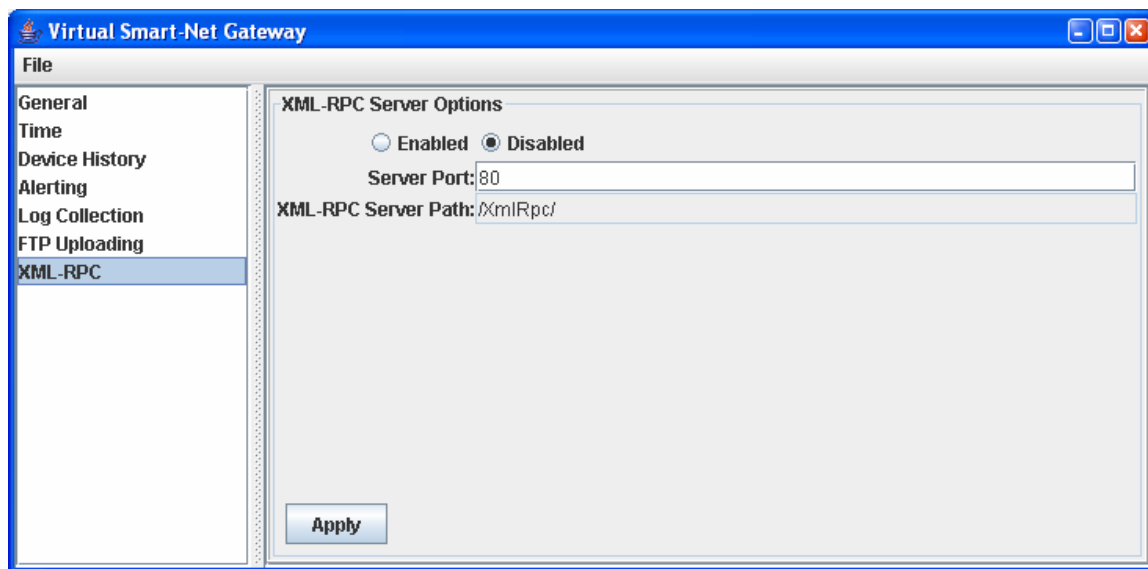
## Testing

To ensure the settings work, a “Test” option is provided. This will create a text file named “<prefix>-Test.txt” (where “<prefix>” is the Log Filename ) containing a log line showing the current date and time and the string “SmartNet Test file”. This file will then be immediately uploaded to the FTP Server.

**Note:** The test button will save the changes and upload a test file regardless of whether it is enabled or disabled.

## XML-RPC Server Options

Click “XML-RPC” on the left to change options for the Virtual Smart-Net Gateway XML-RPC Server (Figure 11). When the changes are made, click **Apply** to save and activate the changes.



**Figure 11.** XML-RPC Server Options

For information on the XML-RPC protocol and methods, refer to the section XML-RPC below.

## Enabling/Disabling

To enable the XML-RPC server, choose “Enabled”. Otherwise, choose “Disabled” to disable the XML-RPC Server. By default, this option is disabled.

## Server Port

Enter the port number for the XML-RPC http server to listen on. By default, this is port 80. If this port is already in use on your system or you don't have permission to use this port, enter a new port (such as 6081). Note that you will have to reconfigure your client software to use the new port.

## XML-RPC Server Path

This is the path on the http server that listens for XML-RPC requests. By default this is “/XmlRpc/”, meaning requests sent to the http server with the url “http://localhost/XmlRpc/” (where localhost is the hostname or IP address of the Virtual Smart-Net Gateway) will be treated as XML-RPC requests. This path is case sensitive. This option is for informative purposes and can not be changed from its default. Your client software must be configured to use this path.

## XML-RPC

### Overview

The Smart-Net Gateway features an XML-RPC server that accepts commands from remote systems. Issuing XML-RPC commands to the Smart-Net Gateway provides a simple and platform independent way to remotely manage the Smart-Net Gateway and devices on the Smart-Net Network.

XML-RPC (<http://www.xml-rpc.com/>) was designed by UserLand Software (<http://www.userland.com/>) in an effort to ease cross-platform remote procedure calls. XML-RPC client implementations are available at the XML-RPC website.

XML-RPC uses the Extended Markup Language (XML) to execute Remote Procedure Calls (RPC) using the HyperText Transfer Protocol (HTTP). Because XML-RPC relies on XML for describing its procedure calls, its data can be read on any platform that is capable of parsing XML documents. Additionally, since it uses HTTP as its transport protocol, it can easily pass through most firewalls and is already implemented on many platforms. Combined, this makes it easy to implement XML-RPC support to nearly any platform that supports internet access and XML.

The descriptions for the XML-RPC Requests in this section are documented with the following notation:

<> denotes an array

Array elements are separated by a comma. Three dots (...) indicates that more elements that follow the same pattern can be present. A vertical pipe symbol (‘|’) indicates that either the element to the left of or to the right of (not both) the ‘|’ symbol may be specified.

{ } denotes a struct

Structure members are separated by a comma. Each name/value pair is separated with an ‘=’ (equal) sign.

[] denotes an optional argument

Square brackets indicate that the enclosed data is optional. It may be an optional parameter to a method or it may be a response that is not received in some situations.

() indicates the data type

Possible data types are: integer, double, boolean, string, and date.

## ***Accessing XML-RPC Methods***

The Virtual Smart-Net Gateway's XML-RPC server, by default, listens to the /XmlRpc/ subdirectory of the web server on port 80 when it is enabled. To access the XML-RPC Methods described below, send a call to the URL: **http://xxx.xxx.xxx.xxx:80/XmlRpc/** where "xxx.xxx.xxx.xxx" is the IP Address or hostname of the computer the Virtual Smart-Net Gateway is running on, and "80" is the port number it is configured to listen on. Within a few seconds, the Smart-Net Gateway will send a response, depending on the method called.

## ***Common Field Descriptions***

Several of the XML-RPC methods described below share the same fields between each other. For brevity, the descriptions of these fields are provided here instead of duplicated in the descriptions of each XML-RPC method.

### **Timestamp**

The "Timestamp" (integer) field represents the time of the Smart-Net Gateway at the time of the method request. This field is an integer value of the number of seconds since January 1, 1970 at 00:00:00. Generally, this value will normally be expressed in Universal Time Coordinated (UTC) or Greenwich Mean Time (GMT).

### **Status**

The "Status" (integer) field is returned for all methods that require Smart-Net communications. This field is a series of bits used as flags to help troubleshoot Smart-Net or protocol problems. If this field is zero (0), this indicates that no Smart-Net problems occurred during the method call. For any other value, convert the integer to a 32-bit binary number and observe the bits that are set (logical 1). Bits 0-7 are related to Smart-Net communications. Bits 8-15 are related to the XML-RPC methods in general. Bits 24-31 depend on the specific method called. If a specific method makes use of bits 24-31, a description will be provided under that method's description.

<b>Bit Number</b>	<b>Error Description</b>
0	Internal Error Occurred
1	Smart-Net Port Adapter was not detected
2	Smart-Net is shorted or improperly wired
3	Communications, device, or Signal Error on Smart-Net
4	Requested Device/Parent was not found
5	Requested Device/Parent is of the wrong type or was not a Parent

6	(Reserved)
7	(Reserved)
8	Invalid parameter(s) supplied
9	Too many parameters supplied
10	Too few parameters supplied
11	Required parameter not supplied
12	(Reserved)
13	(Reserved)
14	(Reserved)
15	(Reserved)

The Bit Number referred to in the table is counted from the least significant bit (0) to the most significant bit (7). For example, if the integer returned is “4”, its binary representation would be: 0000 0000 0000 0100. Since bit number 2 is set, it is determined that the Smart-Net is shorted or improperly wired.

## ROMID

The “ROMID” field (string) is the Smart-Net identification string that is unique to every Smart-Net device. Knowing the ROMID is the only way to read or reconfigure a Smart-Net device. If the ROMID is not known, the **SmartNet.Find** method can be used to discover devices on the network.

## ParentID

The “ParentID” field (string) is the ROMID of a parent device that contains another device. The only way to access a device that is behind a parent device is to know both the Smart-Net identification string of the device (ROMID) and the Smart-Net identification string of its parent device (ParentID).

For example, assume there is a Blade Meter with a ROMID of “2B00000005E0861D-2” connected to a socket of a SmartPDU with a ROMID of “BF0000000101741F”. To access the Blade Meter, the ROMID field would have to be “2B00000005E0861D-2” and the ParentID field would be “BF0000000101741F”.

The **SmartNet.Find** method can be used to discover devices and their parent’s on the network.

## UserID

The “UserID” field (string) is a user-programmable value that can be stored in most Smart-Net devices. The maximum length of this field depends on the physical capabilities of the Smart-Net device being accessed. The UserID stored in a device can be used, for example, to help identify the physical location of a device or the name of a server that the device is attached to. The UserID is not a replacement for knowing the ROMID or a ParentID in the XML-RPC Methods.

## Smart-Net Gateway XML-RPC Methods

### SmartNet.Status

Parameters: <>

Response: { Timestamp=(integer), [FtpLogFileSize=(integer)], [Uptime=(integer)] }

Field Descriptions:

- **FtpLogFileSize** – An integer representing the current size of the log files used by the FTP Logging feature, in bytes.
- **Uptime** – An integer representing the number of seconds since the Smart-Net gateway has been rebooted.

The SmartNet.Status method returns general status information on the Virtual Smart-Net Gateway. The optional fields in the response structure might not be returned if the information is not applicable to the Smart-Net Gateway's current configuration. This method only takes an empty vector for parameters.

### SmartNet.Time

Parameters: <[Timestamp(integer) | NTPServer(string)]>

Response: Timestamp(integer)

Field Descriptions:

- **Timestamp** – An integer representing the number of seconds elapsed since midnight on January 1, 1970.
- **NTPServer** – A string value of the IP Address or domain name for a Network Time Protocol (NTP) server to obtain the time from.

The SmartNet.Time method will set the time on the Smart-Net Gateway. The method will take an integer parameter, a string parameter, or an empty array.

When supplied with an integer, the real time clock on the Smart Net IP Gateway is updated according to the number of seconds since January 1, 1970. It is recommended that the supplied time be in Universal Time Coordinated (UTC).

When supplied with a string value representing the host name or IP Address of a Network Time Protocol (NTP) server, the Smart-Net gateway will retrieve the time from that server. If a connection failed to the NTP server, the time will not get updated and the time returned will represent current time of the Smart-Net Gateway.

In both cases, and when an empty vector is supplied, an integer value representing the number of seconds since 1970 is returned.

### SmartNet.Find

Parameters: <[{{DeviceType=(string)}, [ParentID=(string)]}, ... ]>

Response: <Timestamp (integer), Status (integer), [{{ROMID=(string)}, [ParentID=(string)], DeviceType=(string)}, ... ]>

Field Descriptions:

- **DeviceType** – The type of device (i.e. “SmartWatt” or “SmartSenseTH”).

Status Bits:

Bit Number	Description
24	Unrecognized DeviceType Specified

The SmartNet.Find method is used to discover what devices are connected to the Smart-Net network. Different combinations of parameters may be supplied to this method to determine how to conduct the search on the network.

If an empty vector is supplied, or a struct with no parameters, all Smart-Net Devices on the network are returned **except devices behind a parent** (i.e. Blade Meters inside a SmartPDU).

If a struct with the ParentID field is supplied, only Smart-Net Devices contained within that parent are returned.

If a struct with the DeviceType field is supplied, only devices of that type will be returned (all other types will be filtered).

If a struct with both a ParentID and DeviceType field is supplied, then only devices that match the supplied DeviceType behind the parent will be returned.

Multiple structs can be supplied to search for multiple device types and/or multiple within multiple parents in a single method call.

This method returns a vector with a timestamp and status codes followed by the results of the search as structs. Each result will contain the ROMID of the Smart-Net device found, its device type, and its ParentID (if this device was behind a parent and the request parameters indicated to search behind one or more parents).

The value of DeviceType field is the same that would be used to read or write to a device (as described in this document). For example, if the DeviceType for a device is “SmartWatt”, the XML-RPC method to read that device would simply be “SmartWatt.Read”.

### SmartSenseTH.Read

Parameters: < ROMID(string) | { ROMID=(string), [ParentID=(string)] }, ... >

Response: <Timestamp (integer), Status (integer),  
 { ROMID=(string), [ParentID=(string)], [UserID=(string)], [TRH=(string)],  
 [Temperature=(double)], [DewPoint=(integer)], [AlertEnabled=(boolean)],  
 [LowTemperature=(double)], [HighTemperature=(double)],  
 [LowTRH=(integer)], [HighTRH=(integer)], [LowDewPoint=(integer)],  
 [HighDewPoint=(integer)], [InvalidKey=(boolean)] }, ... >

Field Descriptions:

- **TRH** – A double value representing the true relative humidity as a percentage.
- **Temperature** – A double value representing the temperature in degrees Celsius.

- **DewPoint** – An integer value representing the dew point in degrees Celsius.
- **AlertEnabled** – A boolean value that indicates if this device is allowed to generate alerts for temperature, humidity, and dew point.
- **LowTRH** – An integer value representing the low warning true relative humidity value stored in the device as a percentage.
- **HighTRH** – An integer value representing the high warning true relative humidity value stored in the device as a percentage.
- **LowTemperature** – A double value representing the low temperature warning value stored in the device in degrees Celsius.
- **HighTemperature** – A double value representing the high temperature warning value stored in the device in degrees Celsius.
- **LowDewPoint** – An integer value representing the low dew point warning value stored in the device in degrees Celsius.
- **HighDewPoint** – An integer value representing the high dew point warning value stored in the device in degrees Celsius.
- **InvalidKey** – If the device has an invalid key, this field will be returned with a true value.

The SmartSenseTH.Read method is used to read the current environmental status from one or more Smart-Sense Temperature or Humidity sensors attached to the Smart-Net network.

Any combination of up to 10 strings and/or structs may be supplied in the parameters vector. To read a sensor, the ROMID of the sensor must be known, and, if necessary, the ParentID. To discover sensors, use the SmartNet.Find method.

The information returned is a vector whose first two elements are always integer values representing the time and a status code, respectively. The third and following elements, if present, contain a struct that contains the detailed information about each SmartSense Temperature/Humidity Sensor that was read. Any fields that are not supported by the SmartSense device will not be returned. Additionally, if the device could not be read for any reason, only the ROMID and ParentID (if supplied) will be returned for that device.

### SmartSenseTH.Write

Parameters: < { ROMID=(string), [ParentID=(string)], [UserID=(string)], [AlertEnabled=(boolean)], [LowTemperature=(double)], [HighTemperature=(double)], [LowTRH=(integer)], [HighTRH=(integer)], [LowDewPoint=(integer)], [HighDewPoint=(integer)] }, ... >

Response: < Status (integer), { ROMID=(string), [ParentID=(string)], Success=(boolean) }, ... >

Field Descriptions:

- **Success** – A boolean that indicates if the device was updated on the network.

- **AlertEnabled** – A boolean value indicating if this device is allowed to generate alerts for temperature, humidity, and dew point.
- **LowTRH** – An integer value representing the low warning true relative humidity value stored in the device as a percentage.
- **HighTRH** – An integer value representing the high warning true relative humidity value stored in the device as a percentage.
- **LowTemperature** – A double value representing the low temperature warning value stored in the device in degrees Celsius.
- **HighTemperature** – A double value representing the high temperature warning value stored in the device in degrees Celsius.
- **LowDewPoint** – An integer value representing the low dew point warning value stored in the device in degrees Celsius.
- **HighDewPoint** – An integer value representing the high dew point warning value stored in the device in degrees Celsius.

The SmartSenseTH.Write method will write data into SmartSense Temperature or Humidity devices.

Each supplied struct in the parameters vector represents a single SmartSense Temperature or Humidity sensor device according to the ROMID and the data to store in that device.

The response is a vector whose first element is an integer value representing the time. The second and following elements, if present, contain a struct for each SmartSense device that was specified in the parameters vector. If the success boolean in the struct was true, it indicates that all the supplied data was successfully saved to the device's memory. Otherwise, if the data could not be saved for any reason, the success boolean will be false. For example, attempting to save humidity or dew point values to a temperature-only sensor will result in a success code of "false".

## SmartWatt.Read

Parameters: < ROMID(string) | { ROMID=(string), [ParentID=(string)], [UpdateHistory=(boolean)] }, ... >

Response: <Timestamp (integer), Status (integer), { ROMID=(string), [ParentID=(string)], [Location=(integer)], [UserID=(string)], [Offset=(integer)], [WattHr=(double)], [PwrCycle=(integer)], [ModelNumber=(string)], [InvalidKey=(boolean)], [AvgWatts=(integer)], [PrevReadTime=(integer)], [PrevWattHr=(double)], [PrevPwrCycle=(integer)], [PeakWatts=(integer)], [PeakTime=(integer)], [AlertLow=(integer)], [AlertHigh=(integer)], [AlertEnabled=(boolean)], [CircuitVoltage=(integer)], [CircuitAmperage=(integer)], [CircuitPhase=(integer)], [CircuitGroup=(integer)], [Resolution=(integer)] }, ... >

Field Descriptions:

- **UpdateHistory** – A boolean that indicates whether or not this request should update the previous reading data stored for XML-RPC in the meter. This value defaults to true if not supplied.
- **Location** – If the meter is contained in a SmartPDU, this integer represents where in the SmartPDU this meter was as of the last SmartPDU.Map request.
- **Offset** – An integer representing the number of seconds elapsed since the Timestamp to determine the precise time when this particular SmartWatt device was read.
- **WattHr** – A floating-point value (double) of the cumulative Watt-Hour counter in the SmartWatt device. This value is always returned in Watt-hours and is automatically adjusted based on the Resolution field.
- **PwrCycle** – An integer value of the cumulative number of power interruptions detected by the SmartWatt device
- **ModelNumber** – A string value that contains the ModelNumber of the SmartWatt device.
- **InvalidKey** – If the device has an invalid key, this field will be returned with a true value.
- **AvgWatts** – An integer in Watts of the average watt consumption from PrevReadTime to Timestamp + Offset (now).
- **PrevReadTime** – An integer indicating the number of seconds since January 1, 1970 that this meter was last read by an XML-RPC request.
- **PrevWattHr** – An integer value indicating the cumulative Watt-Hour counter in the device at the time of PrevReadTime. This value is always returned in Watt-hours and is automatically adjusted based on the Resolution field.
- **PrevPwrCycle** – An integer value indicating the cumulative number of power interruptions detected by the SmartWatt device at the time of PrevReadTime.
- **PeakWatts** – An integer value in Watts of the highest load this meter has seen during XML-RPC reads since the peaks were last cleared.
- **PeakTime** – An integer indicating the number of seconds since January 1, 1970 that PeakWatts was last updated.
- **AlertLow** – An integer indicating the Low Alert setting, in Watts, stored in the SmartWatt device.
- **AlertHigh** – An integer indicating the High Alert setting, in Watts, stored in the SmartWatt device.
- **AlertEnabled** – A boolean value indicating if this device is allowed to generate alerts for Watt calculations and power-cycle events.
- **CircuitVoltage** – A user-programmable field that represents the rated voltage of the circuit this meter is connected to.

- **CircuitAmperage** – A user-programmable field that represents the maximum rated current capabilities that this meter is connected to.
- **CircuitPhase** – A user-programmable field that represents which phase this meter is connected to.
- **CircuitGroup** – A user-programmable field indicating which circuit group this meter is a participant in to allow for multiple meters on individual branches to be grouped automatically by reporting software to provide summary information.
- **Resolution** – An integer representing the current resolution setting of the meter. If this value is not matched to the physical properties of the meter, WattHr, AvgWatts, PrevWattHr, and PeakWatts will not be correct.

The SmartWatt.Read method is used to read power and device information from one or more SmartWatt devices connected to the Smart-Net network.

Any combination of up to 10 strings and/or structs may be specified on the parameters vector. To read a meter, the ROMID of the meter must be known, and, if necessary, the ParentID. To discover meters, use the SmartNet.Find method.

If the UpdateHistory boolean is supplied for a meter to be read and is set to “true”, history information will be saved for this reading. This includes storing the current WattHr readings and the date into the meter’s memory. These stored values will then be returned on the next XML-RPC call as the previous values (i.e. PrevWattHr and PrevReadTime). Additionally, the PeakWatts and PeakTime fields will be updated as necessary. If the boolean is set to false, this history data will not be saved. If unspecified, the history is automatically updated. Note that the history data only affects XML-RPC calls and has no effect on readings saved by the FTP Logging function.

The information returned is a vector whose first two elements are always integer values representing a time and a status code, respectively. The third and following elements, if present, contain a struct that contains detailed information about each SmartWatt device it read.

If a SmartWatt device could not be read for any reason, then a struct for that particular device will be returned with no information except the ROMID and ParentID (if supplied).

The Timestamp, PrevReadTime, and PeakTime fields returned are an integer value representing the number of seconds since January 1, 1970. The Offset value returned in each struct is the number of seconds elapsed since Timestamp it took to read the particular SmartWatt’s Watt-Hour Counter. Therefore, Timestamp + Offset represents the exact time a specific meter was read.

PrevReadTime, PrevWattHr, PrevPwrCycle, PeakWatts, and PeakTime are the only fields specific to XML-RPC and are not affected by the FTP Logging function.

The WattHr and PrevWattHr fields are always returned as a double in Watts, regardless of the Resolution field. If the Resolution field is not set correctly, WattHr and PrevWattHr as well as all watt calculations will be inaccurate.

The Location field is automatically updated by the FTP Logging function. If FTP Logging is not enabled, use the SmartPDU.Map method to redetect and update the location field. If the SmartWatt is not behind a SmartPDU, the Location field will not be present.

CircuitVoltage, CircuitAmperage, CircuitPhase, and CircuitGroup are all user-programmable fields to help profile meters. These values are NOT automatically read nor detected by the meter.

## SmartWatt.Write

Parameters: < { ROMID=(string), [ParentID=(string)], [UserID=(string)], [AlertLow=(integer)], [AlertHigh=(integer)], [AlertEnabled=(boolean)], [CircuitVoltage=(integer)], [CircuitAmperage=(integer)], [CircuitPhase=(integer)], [CircuitGroup=(integer)], [Resolution=(integer)], [ClearPrevReadings=(boolean)], [ClearPeaks=(boolean)] }, ... >

Response: < Status (integer), { ROMID=(string), [ParentID=(string)], Success=(boolean) }, ... >

### Field Descriptions:

- **Success** -- A boolean that indicates if the device was updated on the network.
- **AlertLow** – An integer indicating the Low Alert setting, in Watts, stored in the SmartWatt device.
- **AlertHigh** – An integer indicating the High Alert setting, in Watts, stored in the SmartWatt device.
- **AlertEnabled** – A boolean value indicating if this device is allowed to generate alerts for Watt calculations and power-cycle events.
- **CircuitVoltage** – A user-programmable field that represents the rated voltage of the circuit this meter is connected to.
- **CircuitAmperage** – A user-programmable field that represents the maximum rated current capabilities that this meter is connected to.
- **CircuitPhase** – A user-programmable field that represents which phase this meter is connected to.
- **CircuitGroup** – A user-programmable field indicating which circuit group this meter is a participant in to allow for multiple meters on individual branches to be grouped automatically by reporting software to provide summary information.
- **Resolution** – An integer representing the current resolution setting of the meter. If this value is not matched to the physical properties of the meter, WattHr, AvgWatts, PrevWattHr, and PeakWatts will not be correct.
- **ClearPrevReadings** – A boolean that indicates whether or not to clear the previous reading data from the memory for the device for XML-RPC. If unspecified, this defaults to “false”.

- **ClearPeaks** – A boolean that indicates whether or not to clear the peak values from the memory for the device for XML-RPC. If unspecified, this defaults to “false”.

The SmartWatt.Write method will write data into SmartWatt devices.

Each supplied struct in the parameters vector represents a single SmartWatt device according to the ROMID and ParentID (if necessary) and the data to store in that device.

The response is a vector whose first element is an integer value representing the time. The second and following elements, if present, contain a struct for each SmartWatt device that was specified in the parameters vector. If the success boolean in the struct is true, it indicates that all the supplied data was successfully saved to the device’s memory. Otherwise, if the data could not be saved for any reason, the success boolean will be false.

If ClearPrevReadings is set to true, then previous reading counters in the device memory that is used for average watt consumption values will be cleared. In this case, the next SmartWatt.Read will not display these values.

If ClearPeaks is set to true, then the peak value in the SmartWatt will be cleared. In this case, the next SmartWatt.Read will always set a new peak value.

When setting the Resolution field, it is important to make sure to configure it to match the physical properties of the meter. If the resolution is incorrectly set, all Watt calculations and Watt-hour readings will be inaccurate.

## SmartPDU.Read

Parameters: < ROMID(string), ... >

Response: < Timestamp (integer), Status (integer),  
{ ROMID=(string), [UserID=(string)], [Size=(integer)] }, ... >

Field Descriptions:

- **Size** – An integer that represents the maximum number of individual circuits this SmartPDU supports.

The SmartPDU.Read method is used to read information of one or more SmartPDUs attached to the Smart-Net network. Note that this method does NOT read or search for any sensors or meters contained within the SmartPDU.

Up to 10 strings (ROMIDs) may be supplied in the parameters vector. To read a SmartPDU, the ROMID of the SmartPDU must be known. To discover SmartPDUs, use the SmartNet.Find method.

The information returned is a vector whose first two elements are always integer values representing the time and a status code, respectively. The third and following elements, if present, contain a struct that contains the detailed information about each that was read. If the device could not be read for any reason, only the ROMID will be returned for that device.

The Size field returned is preprogrammed by the factory into the SmartPDU and cannot be changed. The UserID field can be changed with either SmartPDU.Write or SmartSenseTH.Write since the same UserID is shared between both the SmartPDU's internal temperature sensor and the SmartPDU itself.

## SmartPDU.Map

Parameters: < ROMID(string) >

Response: < Timestamp (integer), Status (integer), { ROMID=(string), Location=(integer) }, ... >

Field Descriptions:

- **Location** – An integer representing the position of the meter within the SmartPDU.

The SmartPDU.Map method is used to discover the position and which terminal in the SmartPDU a particular blade meter (SmartWatt) is connected to. This method also updates the memory in the SmartWatt with the new location information, and, if the SmartWatt had no UserID set, writes a default UserID based on this location.

Only one SmartPDU may be mapped at a time. To map devices in a SmartPDU, the ROMID of the SmartPDU must be known. The ROMID of devices within a SmartPDU do not need to be known. To discover SmartPDUs, use the SmartNet.Find method.

The information returned is a vector whose first two elements are always integer values representing the time and status code, respectively. The third and following elements, if present, contain a struct that contains the ROMID and location of each device in the SmartPDU whose position was successfully detected.

It is not necessary to call this method if the FTP Logging function is enabled since the meter's location information will automatically be updated during routine reads. However, if FTP Logging is disabled and meters have been changed in a SmartPDU, or this is a new SmartPDU installation, this method must be called in order to get accurate results when reading the meters.

## SmartPDU.Write

Parameters: < { ROMID=(string), [UserID=(string)] }, ... >

Response: < Status (integer), { ROMID=(string), Success=(boolean) }, ... >

Field Descriptions:

- **Success** – A boolean that indicates if the device was updated on the network.

The SmartPDU.Write method will write data into SmartPDU devices.

Each supplied struct in the parameters vector represents a single SmartPDU device according to the ROMID and the data to store in that device.

The response is a vector whose first element is an integer value representing the time. The second and following elements, if present, contain a struct for each SmartPDU device that was specified in the parameters vector. If the success boolean in the struct

was true, it indicates that all the supplied data was successfully saved to the device's memory. Otherwise, if the data could not be saved for any reason, the success boolean will be false.

Note that setting the UserID of a SmartPDU will also set the UserID of the SmartPDU's built in temperature sensor.

### ***Additional Resources***

For additional information regarding the implementation, benefits, and features of the XML-RPC protocol, consult the following resources:

- [Programming Web Services with XML-RPC](#) by Simon St.Laurent, Joe Johnston, and Edd Dumbill. O'Reilly, 2001. ISBN: 0-596-00119-3.
- The XML-RPC Home Page at <http://www.xml-rpc.com/>.
- The XML-RPC Specification at <http://www.xmlrpc.com/spec>.

## Log Collection

### Overview

The Virtual Smart-Net Gateway has the ability to automatically read information on every Smart-Net device connected to it, store it to a log file, and optionally upload this log file to an FTP server at a specified interval. One advantage of using FTP Logging is there is no requirement to continually communicate with the Virtual Smart-Net Gateway to obtain the latest Smart-Net Device readings and status information. Because all data logging is performed unattended and automatically, FTP Logging can be used in situations where a constant internet connection or direct internet access is not desired or feasible.

### Log File Formats

All log file names generated by the Log Collection feature start with the unique serial number of the Port Adapter for easy identification and end with the .txt file extension as follows:

```
<adapter id>-<type>.txt
```

Where <type> is an indication of what type of log data the file represents (described below) and <adapter id> is the unique serial number of the Smart-Net adapter connected to the system.

The reading log files are stored in a structure designed for easy importing into a database. Most Smart-Net devices create a log file of two types: The static configuration data of each device and the changing reading data of each device. The static configuration log files are replaced on every upload and are usually named as “<devicetype>s.txt” where <devicetype> is the type of Smart-Net device. Similarly, the reading data log file is usually named as “<devicetype>Reads.txt”. For example, a SmartWatt will have log files whose filenames end with “SmartWatts.txt” and “SmartWattReads.txt”.

### Common Fields

Several log files will contain the same types of fields as other log files. For brevity, fields that appear in several log files are described here instead of duplicating the same information in the subsequent sections.

SmartNetID	A globally unique identifier for this particular Smart-Net device (also known as the ROMID).
SmartNetGroupID	An identifier of which Smart-Net branch this Smart-Net device is connected to. This is the same unique address/serial number of the Smart-Net Gateway or the USB/Serial port of the Virtual Smart-Net Gateway.

ParentID	The SmartNetID of this device's parent if the device is contained within or is an accessory of another device (such as a Smart-PDU).
Location	If this device is part of a parent, this field may indicate where in the parent that this device is located and/or reading. For example, a Smart-Watt's location in a Smart-PDU will be mapped to the Terminal/Slot number that the meter is attached to.
UserID	A user-programmable field to provide a "friendly" name for this device. If the UserID stored in the device contains quotation marks or commas, they will be replaced by underscores before being saved to the log file.
DateTime	Indicates the Date and Time that this reading occurred. It is in the format of MM/DD/YYYY HH:MM:SS.
TimeZone	Indicates the GMT offset of the DateTime field. It is in the format of +HH:MM where HH is the number of hours and MM is the number of minutes forward or backward (as indicated by a + or - sign) from GMT time.

Note that all fields given in the descriptions will be present in the log file. However, if a field is not applicable to a specific device, that field will be empty (null).

Each field in a log entry is separated by only a comma (,) character. If the field's data type is to be treated as a string of text, it is enclosed in quotes in the log file. Each log entry is terminated by a carriage-return and a newline.

## SmartSenseTH

Log files generated by Smart-Sense Temperature or Humidity sensors are named "xxxx-SmartSenseTHs.txt" and "xxxx-SmartSenseTHReads.txt", where "xxxx" is the serial number of the Smart-Net Adapter.

### *SmartSenseTHs.txt*

The static configuration log file, SmartSenseTHs.txt, contains the following information for each sensor:

```
SmartNetID, SmartNetGroupID, ParentID, Location, UserID,
AlertTempL, AlertTempH, AlertHumidL, AlertHumidH, AlertDewPointL,
AlertDewPointH, AlertEnabled
```

The field descriptions for SmartSenseTHs.txt are:

Location	This field is empty; Reserved for future use
----------	----------------------------------------------

AlertTempL / AlertTempH	The low or high temperature, in degrees Celsius, the sensor must meet or exceed before the Smart-Net Gateway will send an alert.
AlertHumidL / AlertHumidH	The low or high true relative humidity, in percent (0-100), the sensor must meet or exceed before the Smart-Net Gateway will send an alert. If this is not a humidity sensor, these fields will be empty.
AlertDewPointL / AlertDewPointH	The low or high dew point, in degrees Celsius, the sensor must meet or exceed before the Smart-Net Gateway will send an alert. If this is not a humidity sensor, these fields will be empty.
AlertEnabled	A true or false value that indicates if this device is allowed to generate alerts. When false, all temperature, humidity, and dew point alerts are disabled.

For example, one line of a sensor's configuration may be:

```
"B600000059488826", "425E700344890389", , , "RACK 2001-  
A", 15.0, 25.0, 25, 75, 0, 7, true
```

#### *SmartSenseTHReads.txt*

The reading log file, SmartSenseTHReads.txt, contains the following reading data for each sensor:

```
SmartNetID, DateTime, TimeZone, Temp, Humid, DewPoint
```

The field descriptions for SmartSenseTHReads.txt are:

Temp	The temperature, in degrees Celsius, at the sensor.
Humid	The true relative humidity value, a percentage from 0 to 100, at the sensor. If this is not a humidity sensor, this value will be empty.
DewPoint	The Dew Point, as calculated from the temperature and humidity, in degrees Celsius, at the sensor. If this is not a humidity sensor, this value will be empty.

For example, one line of a sensor's log entry may be:

```
"B600000059488826", 02/14/2006 20:41:44, -08:00, 11.0, 65.9, 4.9
```

## **SmartWatt**

Log files generated by Smart-Watts and Blade Meters are named "xxxx-SmartWatts.txt" and "xxxx-SmartWattReads.txt", where "xxxx" is the serial number of the Smart-Net Adapter.

#### *SmartWatts.txt*

The static configuration log file, SmartWatts.txt, contains the following information for each meter:

```
SmartNetID, SmartNetGroupID, ParentID, Location, UserID, Model,
AlertLow, AlertHigh, Resolution, CircuitVolt, CircuitAmp,
CircuitPhase, CircuitGroup, AlertEnabled
```

The field descriptions for SmartWatts.txt are:

Location	The Terminal number/CT number in the Smart-PDU this meter is attached to. If this meter is not part of a Smart-PDU, this field is empty.
Model	The model-number of this Smart-Watt enclosed in quotes.
AlertLow	The lowest number of Watts this meter can read before the Smart-Net Gateway sends an alert.
AlertHigh	The highest number of Watts this meter can read before the Smart-Net Gateway sends an alert.
Resolution	A look-up table indicating the resolution of this Smart-Watt. Note that all readings are converted to Wh in the log files regardless of the resolution.
CircuitVolt	A user-programmable value to indicate the Voltage of the attached circuit.
CircuitAmp	A user-programmable value to indicate the maximum current (in Amps) that the attached circuit will support.
CircuitPhase	A user-programmable value to indicate which phase this meter is attached to (typically 1, 2, or 3). This is used by reporting software to gather total readings grouped by phase.
CircuitGroup	If this meter is part of a group of meters to measure a multi-phase circuit, this value indicates which group this meter belongs to. This is used by reporting software to aggregate multiple meters into a single “virtual” circuit (i.e. a 3-phase circuit with three meters, one per phase). The groups are unique to each SmartNetGroupID.
AlertEnabled	A true or false value that indicates if this device is allowed to generate alerts. When false, the low and high Watt alerts, as well as power cycle alerts, are disabled.

For example, one line of a Smart-Watt’s configuration may be:

```
"7400000005D9E61D-
1", "425E700344890389", "BF0000000101741F", 13, "RACK 1357", "SD25099-
000-0000", 0, 800, 1, 120, 20, 2, 0, false
```

### *SmartWattReads.txt*

The reading log file, SmartWattReads.txt, contains the following data for each meter:

```
SmartNetID, DateTime, TimeZone, WattHours, PowerCycles, Watts
```

The field descriptions of SmartWattReads.txt are:

WattHours	The cumulative Watt-Hour reading of this meter. Note that this value is always converted to Wh, regardless of the resolution setting.
PowerCycles	The cumulative number of power interruptions this meter has detected. If the meter doesn't support power cycle detection, this field will be empty.
Watts	The calculated load, in Watts, seen by this meter as determined from the previous reading and the current reading. If there was no previous reading to calculate from or the Smart-Net Gateway's date and time was adjusted since the previous read, this field will be empty.

For example, one line of a Smart-Watt's reading data may be:

```
"7400000005D9E61D-1", 02/14/2006 22:34:22, -08:00, 94183.1, , 204
```

## **SmartPDU**

The log file generated by a Smart-PDU is named "xxxx-SmartPDUs.txt", where "xxxx" is the serial number of the Smart-Net Adapter. Since the Smart-PDU itself has no changing data, there is no associated SmartPDU "Reads" file. The data of devices contained within a SmartPDU are in the appropriate SmartWatt and SmartSenseTH log files.

The configuration log file, SmartPDUs.txt, contains the following data for each SmartPDU:

```
SmartNetID, SmartNetGroupID, UserID, Size, SmartSenseID
```

The field descriptions of SmartPDUs.txt are:

Size	The maximum number of circuits this PDU can support if fully populated (i.e. 42).
SmartSenseID	The SmartNetID of the Smart-Sense device built-in to the Smart-PDU. The readings of this sensor are logged in the appropriate SmartSenseTH log file.

For example, one line of a Smart-PDU's configuration data may be:

```
"BF0000000101741F", "425E700344890389", "PANEL0507", 42, "7F00000066D97626"
```

## Smart-Net

The log file generated by the Virtual Smart-Net Gateway itself is named “xxxx-SmartNet.txt”, where xxxx is the serial number of the Smart-Net Adapter. This file is replaced on every FTP upload and is used by automated reading software to identify a Smart-Net Gateway or Virtual Smart-Net Gateway in a log directory.

## Exception

The exception file is named “xxxx-Exception.txt” where xxxx is the serial number of the Smart-Net Adapter.

Many problems or notices that the Virtual Smart-Net Gateway’s Log Collection feature encounters will be saved to the Exception log file.

Exceptions may be generated if there is a problem with the Smart-Net network at the time of a device read, an abnormal problem with the Smart-Net Gateway, or a defective Smart-Net device. Intermittent exceptions may also be generated if there is a wiring problem with the Smart-Net network, including, but not limited to, noise, low-quality cables, long distances, lose connections, adding/removing devices, etc.

The entries in the Exception.txt file have the following format:

```
DateTime, TimeZone, ExceptionType, ExceptionDescription
```

The field descriptions of the Exception.txt file are:

ExceptionType	The kind of exception that has occurred, enclosed in quotes. For example, if there was a problem reading a Smart-Watt, this field will be “Smart-Watt”.
ExceptionDescription	A detailed description of the problem that occurred, enclosed in quotes.

## Test

The test file is named “xxxx-Test.txt” where xxxx is the serial number of the Smart-Net Adapter. This file is uploaded when the “Test” option is selected in the configuration.

The entries in this file have the following format:

```
DateTime, TimeZone, "Smart-Net Gateway Test file"
```

The test file is used to verify that communication with the specified FTP Server can be established and is working.

***Intervals***

The minimum recommended Log Reading Interval is approximately three minutes. This is to ensure an accurate measurement of cumulative watt-hour readings as well as an accurate calculation of average watt consumption between readings.