

# BB4-8422 SNMP Client

## Status

### SNMP Client Error Counts

Dashboard / Protocol Status / SNMP Client Status

Error Counts
Read Data
Write Data
Walk Data

Show  entries Search:

Device Number	Local Name	Peer Name	Message count	Error count	Error code
1	spx-pro	192.168.1.137	0	0	
2	v210	192.168.1.23	0	0	

Showing 1 to 2 of 2 entries Previous **1** Next

This page shows an error summary on a device by device basis. There will be one line per each device configured for access by the SNMP Client. If there is a non-zero error count for a device, you can begin to track down where this error is occurring by viewing the various Data pages for the SNMP Client.

2	v210	192.168.1.23	0	0	
<b>Device Number</b>	<b>Local Name</b>	<b>Peer Name</b>	<b>Message count</b>	<b>Error count</b>	<b>Error code</b>

Queries/Errors: Read [0/0], Write[0/0], Walk[0/0]

Holding your mouse over a device will display additional error count information. The counts are "a/b" where "a" is a count and "b" is an error code. The detailed information here is tallied by rule type.

# SNMP Client Read Data

Dashboard / Protocol Status / SNMP Client Status

Error Counts Read Data Write Data Walk Data

Show 10 entries

Search:

Map ↑↓	Error status ↑↓	Seconds since update ↑↓	Present value ↑↓	Device ↑↓	Remote OID ↑↓	Local object ↑↓
1	No Error	5.23	0	1: spx-pro	1.3.6.1.4.1.3815.1.4.1.1.1.2.1	14 : Object 14
2	No Error	5.26	0	2: v210	1.3.6.1.4.1.3815.1.4.1.1.1.2.2	15 : Object 15
3	No Error	5.29	1100	3	1.3.6.1.4.1.3815.1.4.1.1.1.2.6	16 : Object 16
4	No Error	5.3	0	4	1.3.6.1.4.1.3815.1.4.1.1.1.2.3	17 : Object 17
5	No Error	5.3	4	5	1.3.6.1.4.1.3815.1.4.1.1.1.2.4	18 : Object 18
6	No Error	5.32	500	6	1.3.6.1.4.1.3815.1.4.1.1.1.2.5	19 : Object 19
Map	Error status	Seconds since update	Present value	Device	Remote OID	Local object

Showing 1 to 6 of 6 entries

Previous 1 Next

This diagnostic page shows you the most recent data obtained or acted on by the respective map. Most importantly, it shows error indications on a map by map basis. Past error indications can be cleared by clicking the Clear button at the bottom of the page.

# SNMP Client Write Data

Dashboard / Protocol Status / SNMP Client Status

Error Counts Read Data Write Data Walk Data

Show 10 entries Search:

Map ↑↓	Error status ↑↓	Seconds since update ↑↓	Present value ↑↓	Device ↑↓	Remote register address ↑↓	Local object ↑↓
1	No Error	5.37	10	1: spx-pro	1.3.6.1.4.1.3815.1.4.1.1.1.2.1	21 : Object 14
2	No Error	5.38	10	2: v210	1.3.6.1.4.1.3815.1.4.1.1.1.2.5	20 : Object 15
3	No Error	5.38	10	1: spx-pro	1.3.6.1.4.1.3815.1.4.1.1.1.2.2	21 : Object 16
4	No Error	5.39	10	2: v210	1.3.6.1.4.1.3815.1.4.1.1.1.2.6	21 : Object 17
Map	Error status	Seconds since update	Present value	Device	Remote register address	Local object

Showing 1 to 4 of 4 entries Previous 1 Next

Clear write errors

This diagnostic page shows you the most recent data obtained or acted on by the respective map. Most importantly, it shows error indications on a map by map basis. Past error indications can be cleared by clicking the Clear button at the bottom of the page.

## SNMP Client Walk Data

Dashboard / Protocol Status / SNMP Client Status

Error Counts Read Data Write Data Walk Data

Show 10 entries Search:

Rule ↑↓	Device ↑↓	OID ↑↓	Seconds since update ↑↓	Error status ↑↓	Message count ↑↓	Error count ↑↓
1	1	1.3.6.1.4.1.3815.1.4.1.1.1.2.1	0.47	No Error	105	0
Rule	Device	OID	Seconds since update	Error status	Message count	Error count

Showing 1 to 1 of 1 entries Previous 1 Next

Clear walk errors

This diagnostic page shows you the most recent data obtained or acted on by the respective rule. Most importantly, it shows error indications on a rule by rule basis. Past error indications can be cleared by clicking the Clear button at the bottom of the page.

# Configuration

Dashboard / Protocol Configuration / SNMP Client

Read maps

Write maps

Walk Rules

Devices

Config File

Select SNMP Client from the sidebar menu to access SNMP Client Configuration. The SNMP Client Configuration page will have the tabs illustrated above. Click on the tabs to navigate to the areas of interest which are described in detail at the links below.















You may edit individual maps, rules, or devices by first navigating to the tab/page listing those items. Then click the edit icon (pencil). This will take you to the respective "Edit" page listed below.

## SNMP Client Read Maps

Read maps Write maps Walk Rules Devices Config File

Show 10 entries

Search:

Map ↑↓	Device ↑↓	OID ↑↓	Local Object ↑↓
1  	1	1.3.6.1.4.1.318.1.1.1.3.2.1.0	14 : Object 14
2  	2	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.1	15 : Object 15
3  	3	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.2	16 : Object 16
4  	4	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.3	17 : Object 17
5  	5	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.4	18 : Object 18
6  	6	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.1	19 : Object 19
7  	7	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.2	20 : Object 20
Map	Device	OID	Local Object

Showing 1 to 7 of 7 entries

Previous 1 Next

Add 1 new map(s) before map number: 1

SNMP Client Read Maps are where you configure the device to query other SNMP devices for data, and store that data in local data objects. This page shows a list of currently configured maps.

Map ↑↓	Device ↑↓	OID ↑↓	Local Object ↑↓
1  	1	1.3.6.1.4.1.318.1.1.1.3.2.1.0	14 : Object 14

- Click on the map number or pencil icon to jump to the map editing page for this map.
- Click on the trash can icon to delete this map.

Add  new map(s) before map number:

To add or insert new maps, enter a number of maps to add, and select a starting point. Then click the Insert button.



The Set Source Info button is a diagnostic tool. You do not want multiple read maps (from all protocols combined) trying to store data into the same local object. Doing so would result in erroneous data (data would jump back and forth showing the most recently read data from any source). If you only have one source of data, then this isn't an issue. But if there will be multiple sources of data, then use the Set Source Info button to "claim" objects for this protocol function (e.g. SNMP Read).

If objects have already been claimed in another protocol and you try to claim the same objects, you will see one or more error messages displayed at the top of the screen. Go to the Data Objects -> Object Status page to see where objects have been claimed. The Clear Source Info button is also found on the Data Objects -> Object Status page. To clear the claims and try again, use that button. Clicking the Set Source Info button more than once will also result in error messages since objects have already been claimed once.

## SNMP Client Read Map Edit

SNMP Client Read Maps are where you configure the device to query other SNMP devices for data, and store that data in local data objects. This page is where you enter the various parameters to make that happen.

Dashboard / Protocol Configuration / SNMP Client / Read Map #1

### SNMP Client Read Map #1

**Map Number** – Used as a reference in the map list for ordering the maps. Polling is done in round robin fashion in the order of map number.

Read  From

**OID** – A dot field character string specifying the OID to be read from the remote SNMP device using a Get request. The string must have only digits and periods and no embedded spaces.

**Device** – Identifies which device from the device list this map applies to, i.e., determines which IP address to try to establish a connection with.

Applying this hint to octet strings:

Hint: 0 - Character

**Hint** – Provides direction on how to interpret data in the event the value type is Octet String. All ASN data types are interpreted according to their type; however, Octet String will default to being interpreted as a character string unless this hint says otherwise. The exceptions are hint="1" to interpret 4-octet Octet Strings as 32-bit floating point per RFC 6340, and hint="2" to interpret 8-octet Octet Strings as 64-bit floating point per RFC 6340. If the hint is non-zero but the Octet String length is not 4 or 8, then it will still be treated as a character string.

After reading, apply (optional):

Scale: 0      Offset: 0

**Scale** – Provides a scale factor if non-zero (has the effect of being 1 if zero). Data received from a remote SNMP device is multiplied by this factor (if non-zero) before being placed in the local object. Applies to numeric values and numeric local objects only.

**Offset** – Provides an offset to work in conjunction with scale factor. This value will be added to the data value received from a remote SNMP device before being placed in the local object. Applies to numeric values and numeric local objects only.

Poll every 10 seconds

**Poll** – Real poll time in seconds, can be fractional. The sets the rate at which the remote SNMP device will be polled. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again.

Saving result in Local object 14 named Object 14

**Local Object** – Specifies which local object the result of this read operation should be placed in.

Apply Default value: 0 after 0 read failure(s)

**Default Value** – Provides the default value that the local object should be set to in the event the maxfail/failCount is exceeded.

**Max Fail Count** – Optional, provides a count of read failures, if non-zero, that can occur before the local object will be set to the default value given in this map. If zero, the default value will never be applied. If 1, then the default value will be applied upon the first failure (probably not recommended), and so on. The count is reset by a successful read.

Read when  equals

**Index Object** – Optional, allows for selectively enabling this read operation. If an index object (local object number) is given, and its value matches the index value, then this read operation will take place. If an index object is given but the local object’s value does not match the index value, then this read operation will be skipped.

**Index Value** – Optional, used in conjunction with Index Object (see note above).









## SNMP Client Write Maps

Dashboard / Protocol Configuration / SNMP Client

Read maps Write maps Walk Rules Devices Config File

Show 10 entries

Search:

Map	Device	OID	Offset
1  	1	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.1	21 : Object 21
2  	2	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.5	20 : Object 20
3  	3	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.2	21 : Object 21
4  	4	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.6	21 : Object 21
Map	Device	OID	Offset

Showing 1 to 4 of 4 entries

Previous **1** Next

Add  new map(s) before map number:



SNMP Client Write Maps are where you configure the device to send data to other SNMP devices for data. Data sent to the other SNMP devices is taken from the local data objects. This page shows a list of currently configured maps.

Map	Device	OID	Offset
1  	1	1.3.6.1.4.1.3815.1.4.1.1.1.2.1	21 : Object 21

- Click on the map number or pencil icon to jump to the map editing page for this map.
- Click on the trash can icon to delete this map.

Add  new map(s) before map number:

To add or insert new maps, enter a number of maps to add, and select a starting point. Then click the Insert button.

## SNMP Client Write Map Edit

SNMP Client Write Maps are where you configure the device to send data to other SNMP devices for data. Data sent to the other SNMP devices is taken from the local data objects. This page shows a list of currently configured maps.

Dashboard / Protocol Configuration / SNMP Client / Write Map #1

### SNMP Client Write Map #1

**Map Number** – Used as a reference in the map list for ordering the maps. Polling is done in round robin fashion in the order of map number.

Take data from  named

**Local Object** – Specifies the local object number that contains the data that should be sent by this write map.

Apply (optional):

**Scale** – Provides a scale factor if non-zero (has the effect of being 1 if zero). Data to be written is retrieved from the local object and then multiplied by this scale factor before being sent to the remote SNMP device. Applies to numeric values and numeric local objects only.

**Offset** – Provides an offset to work in conjunction with scale factor. This value is added to the value retrieved from the local object (after being multiplied by scale) before being sent to the remote SNMP device. Applies to numeric values and numeric local objects only.

Write   To

**OID** – A dot field character string specifying the OID to be written in the remote SNMP device using a Set request. The string must have only digits and periods and no embedded spaces.

**Device** – Identifies which device from the device list this map applies to, i.e., determines which IP address to try to establish a connection with.

Using ASN Type

**ASN** – Specifies how data should be formatted in the Set request. Select the ASN type from the drop-down list. The following selections are available:

ASN select	ASN type used	Hint assumed	Description
1	ASN Integer	0	32-bit signed integer
2	ASN Timeticks	0	32-bit unsigned integer
3	ASN Gauge	0	32-bit unsigned integer
4	ASN Counter	0	32-bit unsigned integer
5	ASN Octet String	0	String of 8-bit bytes
6	ASN Counter64	0	64-bit unsigned integer
7	ASN Opaque Float	0	Net-SNMP floating point
8	ASN Opaque Double	0	Net-SNMP floating point
9	ASN Opaque U64	0	64-bit unsigned integer
10	ASN Opaque I64	0	64-bit signed integer
11	ASN Opaque Counter64	0	64-bit unsigned integer
12	ASN Bit String	0	Bit string per standard ASN
13	ASN Object ID	0	Object ID per standard ASN
14	ASN IP Address	0	IP address per standard ASN
15	ASN Octet String	1 (RFC 6340 Float)	RFC 6340 32-bit IEEE 754
16	ASN Octet String	2 (RFC 6340 Double)	RFC 6340 64-bit IEEE 754

- Repeat periodically
- Write when object changes
- Enable Max. quiet time

There are three optional qualifiers for writing to another SNMP device. Uncheck the box to disable, and check to enable. When enabled, additional windows will appear to further define that option.

You have the option of writing periodically, writing only when the local object's value has changed by a given margin, or both.

Repeat periodically every

When selected, periodic writing or "polling" will be performed.

**Poll Time** - Real poll time in seconds, can be fractional. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again. The sets the rate at which the remote SNMP device will be updated, provided "Send Periodic" has been enabled. This poll time will be disregarded if Send Periodic is not enabled.

Write when object changes by

When selected, the OID specified in this write map will be written when the local object changes by the value given here.

**Delta** – Specifies the margin by which the local object value should change before sending another Set request to send the new value to a remote SNMP device. Once the changed value has been sent, the new local value is retained for future comparison in determining subsequent additional change. The delta value is disregarded if Send Delta is not enabled. The delta value may be zero, in which case any update to the local object will trigger a write or SNMP Set.

**Minimum Quiet Time** – Real time in seconds, can be fractional. This specifies the minimum amount of time that should elapse between sending of Set requests for this write map. The minimum quiet time has the effect of throttling network traffic.

Enable Max. quiet time of

When selected, this results in writing (SNMP Set) at a minimum interval regardless of delta. This setting would normally be used only in conjunction with writing when value changes.

**Max Quiet Time** – Real time in seconds, can be fractional. If a Set request has not been made either as a result of poll timing or value changing by delta within this time period, then a Set request will be made anyway. This specifies the maximum amount of time that should expire without any Set request being made for any reason.

Write when Local object: 0 equals Value: 0

**Index Object** – Optional, allows for selectively enabling this write operation. If an index object (local object number) is given, and its value matches the index value, then this write operation will take place. If an index object is given but the local object’s value does not match the index value, then this write operation will be skipped.

**Index Value** – Optional, used in conjunction with Index Object (see note above).



## SNMP Client Walk Rules

Dashboard / Protocol Configuration / SNMP Client

Read maps Write maps Walk Rules Devices Config File

Show 10 entries

Search:

Rule	Device	OID	Starting Local Object
1  	1	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.1	12 : Object 12
2  	2	1.3.6.1.4.1.3815.1.4.1.1.1.1.2.8	18 : Object 18
Rule	Device	OID	Starting Local Object

Showing 1 to 2 of 2 entries

Previous 1 Next

Add 1 new rule(s) before rule number: 1 Add walk rule

Set Source Info

SNMP Client Walk Rules are where you configure the device to query other SNMP devices for data by walking a table in that device, and store that data in local data objects. This page shows a list of currently configured rules. The "table walk" will consist of a series of SNMP Get-Next requests resulting in retrieving multiple data values with one walk rule.

Rule	Device	OID	Starting Local Object
1  	1	1.3.6.1.4.1.3815.1.4.1.1.1.2.1	12 : Object 12

- Click on the rule number or pencil icon to jump to the rule editing page for this rule.
- Click on the trash can icon to delete this rule.

Add  new rule(s) before rule number:  [Add walk rule](#)

To add or insert new rules, enter a number of rules to add, and select a starting point. Then click the Add button.



The Set Source Info button is a diagnostic tool. You do not want multiple client functions (from all protocols combined) trying to store data into the same local object. Doing so would result in erroneous data (data would jump back and forth showing the most recently read data from any source). If you only have one source of data, then this isn't an issue. But if there will be multiple sources of data, then use the Set Source Info button to "claim" objects for this protocol function (e.g. SNMP Walk).

If objects have already been claimed in another protocol and you try to claim the same objects, you will see one or more error messages displayed at the top of the screen. Go to the Data Objects -> Object Status page to see where objects have been claimed. The Clear Source Info button is also found on the Data Objects -> Object Status page. To clear the claims and try again, use that button. Clicking the Set Source Info button more than once will also result in error messages since objects have already been claimed once.

## SNMP Client Walk Rule Edit

SNMP Client Walk Rules are where you configure the device to query other SNMP devices for data by walking a table in that device, and store that data in local data objects. This page is where you set up a table walk rule. The "table walk" will consist of a series of SNMP Get-Next requests resulting in retrieving multiple data values with one walk rule.

## SNMP Client Walk Rule #1

**Rule Number** – Used as a reference in the rule list for ordering the rules. Polling is done in round robin fashion in the order of rule number.

Walk table at   From

**OID** – A dot field character string specifying the starting OID for the table walk. The table is walked using GetNext requests, therefore the OID must specify an OID preceding the first OID that one wishes to read.

The table OID is different than OIDs for read and write maps. The table OID includes optional wildcard fields when the walk method is Index or Mask. For example, to walk the alarm table of a UPS using RFC 1628, the OID would be 1.3.6.1.2.1.33.1.6.2.1.\*(2)\*. The \*(2) means allow any value in the second to last field but only act on the value when this field is 2. The last asterisk means disregard the last field entirely (which for RFC 1628 increments with each new alarm until reaching some rollover point, then starts back at 1 again).

**Device** – Identifies which device from the device list this rule applies to, i.e., determines which IP address to try to establish a connection with.

Using    and

**Method** – Defines the method of table walk to be used with values as follows:

- Normal (0)
- Sparse (1)
- Wildcard (2)
- Index (3)
- Mask (4)

The table walking process can become complicated by the fact that tables are allowed to have missing rows, or rows may have intermittent missing columns. In some applications, rows will be populated only temporarily and then they will disappear. The table walk method identified as "Normal" expects the given column to exist in all rows through the range of rows defined by the starting OID and the count. The remaining methods accommodate the other complications permitted by SNMP.

**Method "Normal"** will simply produce a 1 to 1 correlation between table entries and object numbers, placing successive values in successive objects. Data will be interpreted according to the data hint if an octet string is returned, otherwise the ASN encoding will take precedence. If the Get-Next sequence fails to return enough OIDs to fill the 'count' criteria, an error code is set for the device indicating that the table came up short on data.

**Method "Sparse"** is the same as Normal, except missing OIDs in the sequence is anticipated, and the corresponding local objects in the sequence are skipped over if the respective OID is not included in the Get-Next sequence. No error is flagged for table being short on data.


**Method "Wildcard"** allows wildcard fields in the table OID. The walk does not care about order of OIDs returned by Get-Next as long as they match the OID given after discounting wildcard fields. No attempt is made to sequentially pair OIDs with objects. The next 'count' OIDs that successfully match the OID with wildcards will fill the next 'count' of objects beginning with the starting local object number. The OID sent out in the first Get-Next request will have zero in any wildcard fields, and each successive Get-Next will send out the OID from the response to the previous Get-Next.

**Method "Index"** will walk the table, but expect to find that values are OIDs. In other words, the table name is an OID, but the contents of the variable at that name will be another OID. The result is that the OID index (last field of OID) from the value will be used as the offset to calculate which object number is to be affected, and that object will be set to 1 indicating this OID is present in the table. This seemingly odd means of table walking is required in order to translate the alarm table from RFC 1628 for UPS systems into indexable local objects that indicate the presence or absence of alarms defined in RFC 1628. (Note that the alarm table in RFC 1628 is "sparse" meaning its table entries are only present if an alarm is present, and the table is empty if there are no alarms. One cannot simply query OIDs to determine presence of an alarm. Alarms are implied by presence of a table entry that only exists while the alarm is present. Furthermore, the alarm table is simply a circular buffer of alarm entries, and the table index means nothing.)

The object set to 1 by the Index method will not be reset to zero by anything in the table walk rule. Use the timeout feature available in the definition of the object itself to set it to a default value after some amount of time, typically a time longer than the rate at which this table is walked. The result will be a object that is set to 1 when the corresponding alarm exists, and automatically reset to zero sometime after the alarm no longer is found in the alarm table.

**Method "Mask"** is a modified version of "Index". It will walk the table in the same manner, but set bits within the same single object corresponding to the OID index field. OID index .1 will be bit 0, index .2 will be bit 1, and so on. In this case, starting object is the only object number affected, and "count" is the number of bits that will be affected in that one object. If the Index method is used and one wishes to recognize the full range of alarms known to RFC 1628, then 24 objects will be consumed. Using the Mask method, only one 32-bit object is used. Furthermore, non-present alarms will have their corresponding bits cleared without any timeout default value. (Timeout and default should not be used in this case since the single object represents as many as 24 individual states.)

**Hint** – Provides direction on how to interpret data in the event the value type is Octet String. All ASN data types are interpreted according to their type; however, Octet String will default to being interpreted as a character string unless this hint says otherwise. The exceptions are hint="1" to interpret 4-octet Octet Strings as 32-bit floating point per RFC 6340, and hint="2" to interpret 8-octet Octet Strings as 64-bit floating point per RFC 6340. If the hint is non-zero but the Octet String length is not 4 or 8, then it will still be treated as a character string.

Repeat every   seconds

**Poll Time** - Real poll time in seconds, can be fractional. The sets the rate at which the table in the remote SNMP device will be walked. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again.

Save results starting at   named

**Start Object** – Specifies the first local object number that may be filled by this table walk, used in conjunction with count (below).

Populate up to   local object(s) from table

**Count** – Specifies the number of local objects, starting at startObject, that may be filled by this table walk. Most table walks are going to result in filling multiple objects with data.

Walk when   equals  

**Index Object** – Optional, allows for selectively enabling this table walk. If an index object (local object number) is given, and its value matches the index value, then this table walk will take place. If an index object is given but the local object's value does not match the index value, then this table walk will be skipped.

**Index Value** – Optional, used in conjunction with Index Object (see note above).

## SNMP Client Devices



Read maps Write maps Walk Rules **Devices** Config File

Show 10 entries

Search:

Device Number	Name	Peer Name
1	apc	192.168.1.20
2	spx-pro	192.168.1.137
3	v210	192.168.1.23
Device Number	Name	Peer Name

Showing 1 to 3 of 3 entries

Previous 1 Next

Add/Edit 1

Select device

The SNMP Devices page is where you set up the list of SNMP agents that this IoTServer will query while acting as an SNMP manager.

Device Number	Name	Peer Name
1	apc	192.168.1.20

- Click on the device number or pencil icon to jump to the device editing page for this device.
- Click on the trash can icon to delete this device.

Add/Edit 1

Select device

To add additional devices to the list, enter a device number to be added, and click Select Device. If the device number already exists, you will simply be editing that device. If the device number did not exist, you will create that device by editing it.

# SNMP Client Device Edit

This page is where you set up the devices that will be polled (SNMP Get, Set, Get-Next) as defined by the read maps, write maps, and table walk rules. The full set of parameters required for each device is set up once here, and then referenced by device number in any number of maps and rules.

Dashboard / Protocol Configuration / SNMP Client / Device #1

## SNMP Client Device #1

**Device Number** – A number ranging from 1 to device table size, and is referenced in read and write maps, and table walk rules, as the device to which the map applies.

Local Name:

**Name** – Simply provides a reference name for use in the web UI.

Peer Name (Domain or IP address):

**Peer Name** – Provides a definition of where on the network to find the device. The peername in simplest form will be an IP address as illustrated in the XML file example above. However, if the network has access to a DNS server and that server is configured in the network settings of the local device, then peername may be any name that can be found via DNS lookup.

SNMP Version:   Default Poll Period:    seconds

**Version** – Specifies what SNMP version should be used to send the SNMP Get or Set, which in turn determines certain aspects of how the message is formatted. Version may be 1, 2, or 3 where 2 means v2c.

**Rate** – Real or integer poll time in seconds, can be fractional. Used as default when read, write, or walk rule does not have a poll time specified.

Timeout:    seconds Retries:

**Timeout** – Integer number of seconds for session timeout.

**Retries** – Number of times that the SNMP engine should automatically retry a Get or Set if it fails.

Max Var Binds: 3	Poll Tolerance: 1	seconds
------------------	-------------------	---------

**Max Var Binds** – Specifies the maximum number of varbinds that may be included in the same request. When multiple values are requested from the same device, the client will attempt to group them into a single SNMP Get or Set for efficiency (less network traffic). The client will group up to this many varbinds into the same request.

**Poll Tolerance** – Real or integer poll tolerance in seconds, can be fractional. Poll tolerance refers to the amount of time overlap permitted when the client is attempting to group varbinds into a single Get or Set. In other words, if a map or rule timeout is not quite zero but is less than the poll tolerance, it will be treated as zero and grouped into the Get or Set that is already under way.

### SNMPv2 Configuration - The following parameters are used only for v1/v2c

#### SNMP V2 Settings

Max Receive Size: 0	Bytes	Max Send Size: 0	Bytes
SNMP v1/v2c Community: oakpark			

**Max Receive Size** – Provides a limit on the maximum message size that should be accepted, defaults to 1500 if omitted (recommended).

**Max Send Size** – Provides a limit on the maximum message size that should be sent, defaults to 1500 if omitted (recommended). The maxVarBinds should be set to 1 if the maxSendSize is reduced.

**Community** – The community string as defined for SNMP v1 and v2c.

## SNMPv3 Configuration - The following parameters are used only for v3

### SNMP V3 Settings

Security Level:	Authentication, but no privacy	▼
User Name:	jimhogenson	
Authentication Type:	MD5	▼
Authentication Phrase:	[REDACTED]	
Privacy Type:	No privacy	▼
Privacy Phrase:	[REDACTED]	

**Security Level** - Sets security level, 1=noAuthNoPriv, 2=authNoPriv, 3=authPriv. Those are the SNMP acronyms meaning (1) no authentication or privacy, (2) authentication required but privacy is not, (3) both authentication and privacy are required. The term “privacy” means encryption.

**User Name** - Sets the SNMP security name, analogous to username in SNMP terms.

**Authentication Type** - Sets the authentication type, may be “NOAUTH”, “MD5”, or “SHA”. It determines how the username (security name) is hashed when transmitted.

**Authentication Phrase** - Sets the authentication phrase, analogous to an SNMP password. **IMPORTANT:** Use a phrase of at least 10 characters. A very short phrase will be rejected by the encryption algorithm and be indicated as an encryption error.

**Privacy Type** - Sets the privacy type, may be “NOPRIV”, “DES”, or “AES”. This determines which encryption algorithm will be used.

**Privacy Phrase** - Sets the privacy phrase which is used as the encryption key. **IMPORTANT:** Use a phrase of at least 10 characters. A very short phrase will be rejected by the encryption algorithm and be indicated as an encryption error.

Note: For sending traps with SNMPv3, you must know the engine ID of the recipient. The SNMPv3 client, however, will query the remote agent to learn engine ID as well as engine time and boots. It will then use this information to do the actual data query.

## SNMP Client Config File

All of your configuration information is stored in an internal database each time you click the Save button on any page where configuration entries may be made. To make configuration portable from one device to another, and for purposes of retaining a backup copy, the configuration information may be exported and imported as XML or CSV files. This page is where your configuration file management takes place.

It is important to note that the XML file saved within any one client/server function will contain the configuration information for only that function. Depending on overall system configuration, a complete backup may involve more than one XML or CSV file.

Configuration File Load/Save

Configuration file:  Refresh

Load XML Config Into Engine Save Current Config Into .xml file

**XML Files:** When an XML file has been selected, click the Load button to clear the configuration database and reload configuration from the given XML file.

Select an existing name to overwrite or enter a new file name, and then click Save to write the current configuration to the file in XML format.

You may type in a new name in the file name window for purposes of saving a new file. If you click the Refresh button, the file name will be restored to the name currently loaded into the client. The name could have been changed by selecting a file from the list below, or by typing in a new name. If the displayed name has not yet been used, then Refresh will restore the file name to what was most recently loaded.

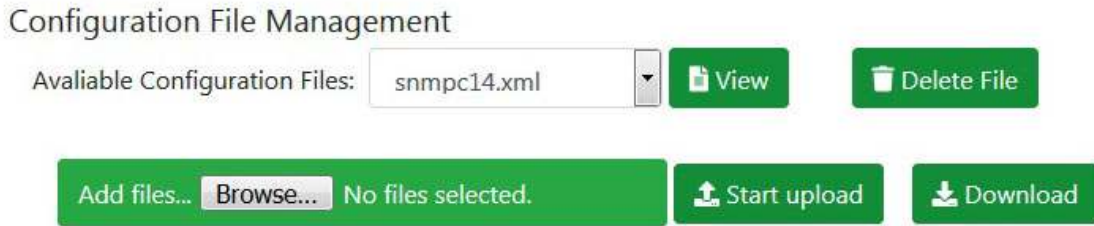
Configuration File Load/Save

Configuration file:  Refresh

Import CSV Config Into Engine Save Current Config Into .csv file

**CSV Files:** If a CSV file is selected, the Load and Save buttons will change into load/save CSV buttons. When a CSV file has been selected, click the Load button to clear the configuration database and reload configuration from the given CSV file.

Select an existing name to overwrite or enter a new file name, and then click Save to write the current configuration to the file in CSV format.



The drop-down list will show a list of all configuration files currently found in the device's configuration folder. When you select an XML or CSV file from this list, the name will be copied to the Load/Save section of this page for pending load or save.

You may view the selected file by simply clicking View. You can delete the file by clicking Delete.

You may upload files to the IoTServer from your PC. Start by clicking Browse, and then use the browser's file dialog to locate the file on your PC. Once a file is selected on your PC, click the "Start upload" button to initiate the transfer.

You may also download files from the IoTServer to your PC. Click the Download button to transfer the selected file to your PC.



Any time an XML or CSV file is loaded, an error log file is generated. The error log file will be given the same name as the loaded file, but with ".err" as the suffix instead of ".xml" or ".csv". You may view the error log by selecting it from the list and clicking View.

Status is normally displayed in a message box at the top of the screen when the load or save operation is complete. But if you want to double check the status of the previous file operation, click Check Status.



Current Status: Running 

The task (client or server) needs to be suspended while a file load operation is in progress to prevent acting on any partial configurations. This suspend/resume operation will normally happen automatically as part of the sequence invoked by the Load button when loading an XML file. The task must be explicitly suspended here for importing a CSV file. The Suspend button will become a Resume button when the task is suspended. Click Resume to continue operation. The current status is always displayed here.

## SNMP Client XML Files

### Example XML File

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<!-- IoT Server SNMP Client configuration file -->

<configuration>

<snmpclient_devices>
<dev id="1" peername="192.168.1.20" name="apc" version="3" maxVarBinds="3"
pollTolerance="1" secLevel="3" username="jim" authType="MD5"
authPhrase="jimsAuthPhrase" privType="DES" privPhrase="jimsPrivPhrase"/>>
<dev id="2" peername="192.168.1.23" name="v210" version="2" community="private"
maxVarBinds="3" pollTolerance="1"/>
</snmpclient_devices>

<snmpclient_read>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.1" dev="1" localobj="14" poll="10.0"/>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.2" dev="1" localobj="15" poll="10.0"/>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.6" dev="2" localobj="17" poll="10.0"/>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.3" dev="1" localobj="16" poll="10.0"/>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.4" dev="2" localobj="18" poll="10.0"/>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.5" dev="2" localobj="19" poll="10.0"/>
</snmpclient_read>

<snmpclient_write>
<map oid="1.3.6.1.4.1.3815.1.4.1.1.1.2.5" dev="2" localobj="20" asn="1" delta="0"/>
</snmpclient_write>

<snmpclient_walk>
<rule oid="1.3.6.1.2.1.33.1.6.2.1.*(2).*" dev="1" count="24" method="4" startobj="14"
```

```
poll="10.0"/>
</snmpclient_walk>

</configuration>
```

## <snmpclient\_devices> section

**id="n"** – A number ranging from 1 to device table size, and is referenced in read and write maps, and table walk rules, as the device to which the map applies.

**peername="xxxx"** – Provides a definition of where on the network to find the device. The peername in simplest form will be an IP address as illustrated in the XML file example above. However, if the network has access to a DNS server and that server is configured in the network settings of the local device, then peername may be any name that can be found via DNS lookup.

**rate="n.nn"** – Real or integer poll time in seconds, can be fractional. Used as default when read, write, or walk rule does not have a poll time specified.

**version="n"** – Specifies what SNMP version should be used to send the SNMP Get or Set, which in turn determines certain aspects of how the message is formatted. Version may be 1, 2, or 3 where 2 means v2c.

**timeout="n"** – Integer number of seconds for session timeout.

**retries="n"** – Number of times that the SNMP engine should automatically retry a Get or Set if it fails.

**maxVarBinds="n"** – Specifies the maximum number of varbinds that may be included in the same request. When multiple values are requested from the same device, the client will attempt to group them into a single SNMP Get or Set for efficiency (less network traffic). The client will group up to this many varbinds into the same request.

**maxRecvSize="n"** – Provides a limit on the maximum message size that should be accepted, defaults to 1500 if omitted (recommended).

**maxSendSize="n"** – Provides a limit on the maximum message size that should be sent, defaults to 1500 if omitted (recommended). The maxVarBinds should be set to 1 if the maxSendSize is reduced.

**pollTolerance="n.nn"** – Real or integer poll tolerance in seconds, can be fractional. Poll tolerance refers to the amount of time overlap permitted when the client is attempting to group varbinds into a single Get or Set. In other words, if a map or rule timeout is not quite zero but is less than the poll tolerance, it will be treated as zero and grouped into the Get or Set that is already under way.

**name="xxxx"** – Simply provides a reference name for use in the web UI.



**community="xxxx"** – The community string as defined for SNMP v1 and v2c.

**secLevel="n"** - Sets security level, 1=noAuthNoPriv, 2=authNoPriv, 3=authPriv. Those are the SNMP acronyms meaning (1) no authentication or privacy, (2) authentication required but privacy is not, (3) both authentication and privacy are required. The term “privacy” means encryption. (Used only for SNMPv3.)

**username="xxxx"** - Sets the SNMP security name, analogous to username in SNMP terms. (Used only for SNMPv3.)

**authType="xxx"** - Sets the authentication type, may be “NOAUTH”, “MD5”, or “SHA”. It determines how the username (security name) is hashed when transmitted. (Used only for SNMPv3.)

**authPhrase="xxx"** - Sets the authentication phrase, analogous to an SNMP password. (Used only for SNMPv3.)

**privType="xxx"** - Sets the privacy type, may be “NOPRIV”, “DES”, or “AES”. This determines which encryption algorithm will be used. (Used only for SNMPv3.)

**privPhrase="xxx"** - Sets the privacy phrase which is used as the encryption key. (Used only for SNMPv3.)

Note: For sending traps with SNMPv3, you must know the engine ID of the recipient. The SNMPv3 client, however, will query the remote agent to learn engine ID as well as engine time and boots. It will then use this information to do the actual data query.

## <snmpclient\_read> section

Map Number is implied by order in XML – Used as a reference in the map list for ordering the maps. Polling is done in round robin fashion in the order of map number.

**oid="n.n.n.n..."** – A dot field character string specifying the OID to be read from the remote SNMP device using a Get request. The string must have only digits and periods and no embedded spaces.

**hint="n"** – Provides direction on how to interpret data in the event the value type is Octet String. All ASN data types are interpreted according to their type; however, Octet String will default to being interpreted as a character string unless this hint says otherwise. The exceptions are hint="1" to interpret 4-octet Octet Strings as 32-bit floating point per RFC 6340, and hint="2" to interpret 8-octet Octet Strings as 64-bit floating point per RFC 6340. If the hint is non-zero but the Octet String length is not 4 or 8, then it will still be treated as a character string.

**dev="n"** – Identifies which device from the device list this map applies to, i.e., determines which IP address to try to establish a connection with.

**scale="n.nn"** – Provides a scale factor if non-zero (has the effect of being 1 if zero). Data received from a remote SNMP device is multiplied by this factor (if non-zero) before being placed in the local object. Applies to numeric values and numeric local objects only.

**offset="n.nn"** – Provides an offset to work in conjunction with scale factor. This value will be added to the data value received from a remote SNMP device before being placed in the local object. Applies to numeric values and numeric local objects only.

**poll="n.nn"** – Real poll time in seconds, can be fractional. The sets the rate at which the remote SNMP device will be polled. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again.

**localObj="n"** – Specifies which local object the result of this read operation should be placed in.

**default="n.nn"** – Provides the default value that the local object should be set to in the event the maxfail/failCount is exceeded.

**maxfail="n"** – Optional, provides a count of read failures, if non-zero, that can occur before the local object will be set to the default value given in this map. If zero, the default value will never be applied. If 1, then the default value will be applied upon the first failure (probably not recommended), and so on. The count is reset by a successful read.

**xobj="n"** – Optional, allows for selectively enabling this read operation. If an index object (local object number) is given, and its value matches the indexValue, then this read operation will take place. If an indexObject is given but the local object's value does not match the indexValue, then this read operation will be skipped.

**xvalue="n"** – Optional, used in conjunction with indexObject (see note above).

### <snmpclient\_write> section

Map Number is implied by order in XML – Used as a reference in the map list for ordering the maps. Polling is done in round robin fashion in the order of map number.

**oid="n.n.n.n...."** – A dot field character string specifying the OID to be written in the remote SNMP device using a Set request. The string must have only digits and periods and no embedded spaces.

**asn="n"** – Specifies how data should be formatted in the Set request. The code given here is a selector, not the actual ASN code. The selection is made from the following table:

ASN select	ASN type used	Hint assumed	Description
------------	---------------	--------------	-------------

## BB4-8422 SNMP Client

1	ASN Integer	0	32-bit signed integer
2	ASN Timeticks	0	32-bit unsigned integer
3	ASN Gauge	0	32-bit unsigned integer
4	ASN Counter	0	32-bit unsigned integer
5	ASN Octet String	0	String of 8-bit bytes
6	ASN Counter64	0	64-bit unsigned integer
7	ASN Opaque Float	0	Net-SNMP floating point
8	ASN Opaque Double	0	Net-SNMP floating point
9	ASN Opaque U64	0	64-bit unsigned integer
10	ASN Opaque I64	0	64-bit signed integer
11	ASN Opaque Counter64	0	64-bit unsigned integer
12	ASN Bit String	0	Bit string per standard ASN
13	ASN Object ID	0	Object ID per standard ASN
14	ASN IP Address	0	IP address per standard ASN
15	ASN Octet String	1 (RFC 6340 Float)	RFC 6340 32-bit IEEE 754
16	ASN Octet String	2 (RFC 6340 Double)	RFC 6340 64-bit IEEE 754

**dev="n"** – Identifies which device from the device list this map applies to, i.e., determines which IP address to try to establish a connection with.

**scale="n.nn"** – Provides a scale factor if non-zero (has the effect of being 1 if zero). Data to be written is retrieved from the local object and then multiplied by this scale factor before being sent to the remote SNMP device. Applies to numeric values and numeric local objects only.

**offset="n.nn"** – Provides an offset to work in conjunction with scale factor. This value is added to the value retrieved from the local object (after being multiplied by scale) before being sent to the remote SNMP device. Applies to numeric values and numeric local objects only.

**poll="n.nn"** - Real poll time in seconds, can be fractional. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again. The sets the rate at which the remote SNMP device will be updated, provided "sendPeriodic" has been enabled. This poll time will be disregarded if

sendPeriodic is not enabled. The Send Periodic feature is automatically enabled by the presence of this tag in the XML file.

**maxquiet="n.nn"** – Real time in seconds, can be fractional. If a Set request has not been made either as a result of poll timing or value changing by delta within this time period, then a Set request will be made anyway. This specifies the maximum amount of time that should expire without any Set request being made for any reason. The Max Quiet Time feature is automatically enabled by the presence of this tag in the XML file.

**minquiet="n.nn"** – Real time in seconds, can be fractional. This specifies the minimum amount of time that should elapse between sending of Set requests for this write map. The minimum quiet time has the effect of throttling network traffic.

**delta="n.nn"** – Specifies the margin by which the local object value should change before sending another Set request to send the new value to a remote SNMP device. Once the changed value has been sent, the new local value is retained for future comparison in determining subsequent additional change. The delta value is disregarded if sendDelta is not enabled. The Send on Delta feature is enabled automatically by the presence of this tag in the XML file.

**localObj="n"** – Specifies the local object number that contains the data that should be sent by this write map.

**xobj="n"** – Optional, allows for selectively enabling this write operation. If an index object (local object number) is given, and its value matches the indexValue, then this write operation will take place. If an indexObject is given but the local object's value does not match the indexValue, then this write operation will be skipped.

**xvalue="n"** – Optional, used in conjunction with indexObject (see note above).

## <snmpclient\_walk> section

Rule Number is implied by order in XML – Used as a reference in the rule list for ordering the maps. Polling is done in round robin fashion in the order of rule number.

**oid="n.n.n.n..."** – A dot field character string specifying the starting OID for the table walk. The table is walked using GetNext requests, therefore the OID must specify an OID preceding the first OID that one wishes to read.

The table OID is different than OIDs for read and write maps. The table OID includes optional wildcard fields when the walk method is Index or Mask. For example, to walk the alarm table of a UPS using RFC 1628, the OID would be 1.3.6.1.2.1.33.1.6.2.1.\*(2).\* The \*(2) means allow any value in the second to last field but only act on the value when this field is 2. The last asterisk means disregard the last field entirely (which for RFC 1628 increments with each new alarm until reaching some rollover point, then starts back at 1 again).

**method="n"** – Defines the method of table walk to be used with values as follows:

- 0 = Normal
- 1 = Sparse
- 2 = Wildcard
- 3 = Index
- 4 = Mask

The table walking process can become complicated by the fact that tables are allowed to have missing rows, or rows may have intermittent missing columns. In some applications, rows will be populated only temporarily and then they will disappear. The table walk method identified as "Normal" expects the given column to exist in all rows through the range of rows defined by the starting OID and the count. The remaining methods accommodate the other complications permitted by SNMP.

Method "Normal" will simply produce a 1 to 1 correlation between table entries and object numbers, placing successive values in successive objects. Data will be interpreted according to the data hint if an octet string is returned, otherwise the ASN encoding will take precedence. If the Get-Next sequence fails to return enough OIDs to fill the 'count' criteria, an error code is set for the device indicating that the table came up short on data.

Method "Sparse" is the same as Normal, except missing OIDs in the sequence is anticipated, and the corresponding local objects in the sequence are skipped over if the respective OID is not included in the Get-Next sequence. No error is flagged for table being short on data.

Method "Wildcard" allows wildcard fields in the table OID. The walk does not care about order of OIDs returned by Get-Next as long as they match the OID given after discounting wildcard fields. No attempt is made to sequentially pair OIDs with objects. The next 'count' OIDs that successfully match the OID with wildcards will fill the next 'count' of objects beginning with the starting local object number. The OID sent out in the first Get-Next request will have zero in any wildcard fields, and each successive Get-Next will send out the OID from the response to the previous Get-Next.

Method "Index" will walk the table, but expect to find that values are OIDs. In other words, the table name is an OID, but the contents of the variable at that name will be another OID. The result is that the OID index (last field of OID) from the value will be used as the offset to calculate which object number is to be affected, and that object will be set to 1 indicating this OID is present in the table. This seemingly odd means of table walking is required in order to translate the alarm table from RFC 1628 for UPS systems into indexable local objects that indicate the presence or absence of alarms defined in RFC 1628. (Note that the alarm table in RFC 1628 is "sparse" meaning its table entries are only present if an alarm is present, and the table is empty if there are no alarms. One cannot simply query OIDs to determine presence of an alarm. Alarms are implied by presence of a table entry that only exists while the alarm is present. Furthermore, the alarm table is simply a circular buffer of alarm entries, and the table index means nothing.)

The object set to 1 by the Index method will not be reset to zero by anything in the table walk rule. Use the timeout feature available in the definition of the object itself to set it to a default value after some amount of time, typically a time longer than the rate at which this table is

walked. The result will be a object that is set to 1 when the corresponding alarm exists, and automatically reset to zero sometime after the alarm no longer is found in the alarm table.

Method "Mask" is a modified version of "Index". It will walk the table in the same manner, but set bits within the same single object corresponding to the OID index field. OID index .1 will be bit 0, index .2 will be bit 1, and so on. In this case, starting object is the only object number affected, and "count" is the number of bits that will be affected in that one object. If the Index method is used and one wishes to recognize the full range of alarms known to RFC 1628, then 24 objects will be consumed. Using the Mask method, only one 32-bit object is used. Furthermore, non-present alarms will have their corresponding bits cleared without any timeout default value. (Timeout and default should not be used in this case since the single object represents as many as 24 individual states.)

**hint="n"** – Provides direction on how to interpret data in the event the value type is Octet String. All ASN data types are interpreted according to their type; however, Octet String will default to being interpreted as a character string unless this hint says otherwise. The exceptions are hint="1" to interpret 4-octet Octet Strings as 32-bit floating point per RFC 6340, and hint="2" to interpret 8-octet Octet Strings as 64-bit floating point per RFC 6340. If the hint is non-zero but the Octet String length is not 4 or 8, then it will still be treated as a character string.

**dev="n"** – Identifies which device from the device list this rule applies to, i.e., determines which IP address to try to establish a connection with.

**poll="n.nn"** - Real poll time in seconds, can be fractional. The sets the rate at which the table in the remote SNMP device will be walked. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again.

**startObj="n"** – Specifies the first local object number that may be filled by this table walk, used in conjunction with count (below).

**count="n"** – Specifies the number of local objects, starting at startObject, that may be filled by this table walk. Most table walks are going to result in filling multiple objects with data.

**xobj="n"** – Optional, allows for selectively enabling this table walk. If an index object (local object number) is given, and its value matches the indexValue, then this table walk will take place. If an indexObject is given but the local object's value does not match the indexValue, then this table walk will be skipped.

**xvalue="n"** – Optional, used in conjunction with indexObject (see note above).

## SNMP Client CSV Files

A CSV file may be imported to configure various aspects of the IoTServer (or Babel Buster gateway). A single CSV file may contain multiple sections. When a file including an “Objects” section is imported by the Data Engine, local objects will be configured. When a file including one or more “Modbus” sections is imported by an instance of the Modbus Engine, Modbus gateway functionality will be configured. The same Modbus file may be imported by a Modbus Client or Modbus Server, and either RTU or TCP, and only those sections of interest to that Modbus function will be imported. The CSV file may also contain one or more SNMP sections, and so forth.

A section begins when the word “Begin” appears in the first column of a line. All lines up to and including a line that begins with the word “End” will be taken to be part of that section.

The line immediately following the “Begin” line must be a header line. A Header line is one which labels the columns of data that will follow the Header line.

All lines following the Header line are data lines that are expected to contain the same number of columns as the Header line, and whose contents are defined by the labels found in each column of the Header line.

Labels in the section Begin and End lines, and labels in the Header line are NOT case sensitive and will be interpreted equally whether upper case, lower case, or some combination of both (for readability).

Labels may NOT contain embedded spaces. A label is terminated by a comma, line-end, or space. Labels may not be encapsulated in quote characters; however, data content in data lines may be encapsulated in quote characters and may contain embedded spaces or blanks if quoted.

Some labels in the Header line may be considered optional. The minimum required columns are indicated in the definition of each data section.

Columns in the Header line do not have to follow any particular order. They may be rearranged to the user’s liking. The only restriction is that data in subsequent data lines must match up with the labels placed in the Header line. Data lines may contain fewer columns than the Header line, but may not contain more. Data columns that the user wishes to deliberately omit, but omit between included columns, should be indicated by place holder commas (which will simply appear as blank cells in a spread sheet program).

A Begin line will contain three columns:

Column 1: BEGIN

Column 2: Function as noted below

Column 3: Sub-function as noted in definition of the Function.

Functions may be any of the following (with this listed expanded from time to time):

- LOCALDATA
- MODBUS
- SNMP

Sub-functions:  
SNMP (client)

- DEVICES
- READMAPS
- WRITEMAPS
- WALKRULES

SNMP (agent)

- MIBVARS
- DEVICES
- TRAPSENDERULES

SNMP (trap receiver)

- TRAPRECVRULES

NOTE: The same SNMP CSV file may NOT contain both client and server sections as DEVICES becomes ambiguous. SNMP requires different applications for different purposes (csiSnmClient, csiSnmAgent, csiSnmTrapRecv). A fourth application, csiTrapHandler, is also associated with csiSnmAgent.

## SNMP Client Example

The following illustrates an SNMP client CSV file. There will typically be only one instance of a client, unless multiple instances are implemented for load sharing purposes.

```
BEGIN,SNMP,DEVICES
NUMBER,PEERNAME,VERSION,COMMUNITY,POLLTIME,TIMEOUT,RETRIES,MAXV
ARBINDS,MAXRECVSIZE,MAXSENDSIZE,POLLTOLERANCE,NAME
1,192.168.1.137,2,birchwood,10.000000,0,0,3,0,0,1.000000,spx-pro
2,192.168.1.23,2,private,10.000000,0,0,3,0,0,1.000000,v210
3,192.168.1.20,2,public,10.000000,0,0,3,0,0,1.000000,apc
END
BEGIN,SNMP,READMAPS
DEVICE,OID,HINT,SCALE,OFFSET,DESTOBJ,POLLTIME,DEFVALUE,FAILCOUNT,IND
EXOBJ,INDEXVAL
1,1.3.6.1.4.1.3815.1.4.1.1.1.2.1,NONE,0.000000,0.000000,1,10.000000,0.000000,0,0,0
1,1.3.6.1.4.1.3815.1.4.1.1.1.2.2,NONE,0.000000,0.000000,2,10.000000,0.000000,0,0,0
2,1.3.6.1.4.1.3815.1.4.1.1.1.2.6,NONE,0.000000,0.000000,3,10.000000,0.000000,0,0,0
1,1.3.6.1.4.1.3815.1.4.1.1.1.2.3,NONE,0.000000,0.000000,4,10.000000,0.000000,0,0,0
```



```

2,1.3.6.1.4.1.3815.1.4.1.1.1.2.4,NONE,0.000000,0.000000,5,10.000000,0.000000,0,0,0
2,1.3.6.1.4.1.3815.1.4.1.1.1.2.5,NONE,0.000000,0.000000,6,10.000000,0.000000,0,0,0
END
BEGIN,SNMP,WRITEMAPS
SOURCEOBJ,SCALE,OFFSET,DEVICE,OID,ASNSELECT,SENDPERIODIC,POLLTIME,SE
NDMAXQUIET,MAXQUIETTIME,SENDONDELTA,DELTA,MINQUIETTIME,INDEXOBJ,
INDEXVAL
20,0.000000,0.000000,2,1.3.6.1.4.1.3815.1.4.1.1.1.2.5,INTEGER,N,0.000000,N,0.000000,Y,0.
000000,0.000000,0,0
END
BEGIN,SNMP,WALKRULES
DEVICE,OID,METHOD,HINT,STARTOBJ,OBJCOUNT,POLLTIME,INDEXOBJ,INDEXVA
L
3,1.3.6.1.2.1.33.1.6.2.1.*(2).*,MASK,NONE,14,24,10.000000,0,0
END

```

## SNMP (client) DEVICES Section

The DEVICES section identifies remote SNMP devices that will be read or written by the IoTServer as directed by read and write maps.

**Number** – A number ranging from 1 to device table size, and is referenced in read and write maps, and table walk rules, as the device to which the map applies.

**PeerName** – Provides a definition of where on the network to find the device. The peername in simplest form will be an IP address as illustrated in the XML file example above. However, if the network has access to a DNS server and that server is configured in the network settings of the local device, then peername may be any name that can be found via DNS lookup.

**Version** – Specifies what SNMP version should be used to send the SNMP Get or Set, which in turn determines certain aspects of how the message is formatted. Version may be 1, 2, or 3 where 2 means v2c.

**Community** – The community string as defined for SNMP v1 and v2c.

**PollTime** – Poll time in seconds, can be fractional. Used as default when read, write, or walk rule does not have a poll time specified.

**Timeout** – Integer number of seconds for session timeout.

**Retries** – Number of times that the SNMP engine should automatically retry a Get or Set if it fails.

**MaxVarBinds** – Specifies the maximum number of varbinds that may be included in the same request. When multiple values are requested from the same device, the client will attempt to

group them into a single SNMP Get or Set for efficiency (less network traffic). The client will group up to this many varbinds into the same request.

**MaxRecvSize** – Provides a limit on the maximum message size that should be accepted, defaults to 1500 if omitted (recommended).

**MaxSendSize** – Provides a limit on the maximum message size that should be sent, defaults to 1500 if omitted (recommended). The maxVarBinds should be set to 1 if the maxSendSize is reduced.

**PollTolerance** – Poll tolerance in seconds, can be fractional. Poll tolerance refers to the amount of time overlap permitted when the client is attempting to group varbinds into a single Get or Set. In other words, if a map or rule timeout is not quite zero but is less than the poll tolerance, it will be treated as zero and grouped into the Get or Set that is already under way.

**Name** – Arbitrary reference name for use in the web UI.

**SecLevel** - Sets security level, 1=noAuthNoPriv, 2=authNoPriv, 3=authPriv. Those are the SNMP acronyms meaning (1) no authentication or privacy, (2) authentication required but privacy is not, (3) both authentication and privacy are required. The term “privacy” means encryption. (Used only for SNMPv3.)

**Username** - Sets the SNMP security name, analogous to username in SNMP terms. (Used only for SNMPv3.)

**AuthType** - Sets the authentication type, may be “NOAUTH”, “MD5”, or “SHA”. It determines how the username (security name) is hashed when transmitted. (Used only for SNMPv3.)

**AuthPhrase** - Sets the authentication phrase, analogous to an SNMP password. (Used only for SNMPv3.)

**PrivType** - Sets the privacy type, may be “NOPRIV”, “DES”, or “AES”. This determines which encryption algorithm will be used. (Used only for SNMPv3.)

**PrivPhrase** - Sets the privacy phrase which is used as the encryption key. (Used only for SNMPv3.)

## SNMP (client) READMAPS Section

**Device** – Identifies which device from the device list this map applies to, i.e., determines which IP address to try to establish a connection with.

**OID** – A dot field character string specifying the OID to be read from the remote SNMP device using a Get request. The string must have only digits and periods and no embedded spaces.

**Hint** – Provides direction on how to interpret data in the event the value type is Octet String. All ASN data types are interpreted according to their type; however, Octet String will default to being interpreted as a character string unless this hint says otherwise, and the length is exactly 4 or 8 bytes (octets).

Hint Label	Description
None	No special interpretation
Float	Treat 4-byte Octet String as RFC 6340 IEEE 754 32-bit floating point
Double	Treat 8-byte Octet String as RFC 6340 IEEE 754 64-bit floating point

**Scale** – Provides a scale factor if non-zero (has the effect of being 1 if zero). Data received from a remote SNMP device is multiplied by this factor (if non-zero) before being placed in the local object. Applies to numeric values and numeric local objects only.

**Offset** – Provides an offset to work in conjunction with scale factor. This value will be added to the data value received from a remote SNMP device before being placed in the local object. Applies to numeric values and numeric local objects only.

**DestObj** – Specifies which local object the result of this read operation should be placed in.

**PollTime** – Poll time in seconds, can be fractional. The sets the rate at which the remote SNMP device will be polled. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again.

**DefValue** – Provides the default value that the local object should be set to in the event the FailCount is exceeded.

**FailCount** – Optional, provides a count of read failures, if non-zero, that can occur before the local object will be set to the default value given in this map. If zero, the default value will never be applied. If 1, then the default value will be applied upon the first failure (probably not recommended), and so on. The count is reset by a successful read.

**IndexObj** – Optional, allows for selectively enabling this read operation. If an index object (local object number) is given, and its value matches the IndexVal value, then this read operation will take place. If an IndexObj is given but the local object's value does not match the IndexVal, then this read operation will be skipped.

**IndexVal** – Optional, used in conjunction with IndexObj (see note above).

## SNMP (client) WRITEMAPS Section

**SourceObj** – Specifies the local object number that contains the data that should be sent by this write map.

**Scale** – Provides a scale factor if non-zero (has the effect of being 1 if zero). Data to be written is retrieved from the local object and then multiplied by this scale factor before being sent to the remote SNMP device. Applies to numeric values and numeric local objects only.

**Offset** – Provides an offset to work in conjunction with scale factor. This value is added to the value retrieved from the local object (after being multiplied by scale) before being sent to the remote SNMP device. Applies to numeric values and numeric local objects only.

**Device** – Identifies which device from the device list this map applies to, i.e., determines which IP address to try to establish a connection with.

**OID** – A dot field character string specifying the OID to be written in the remote SNMP device using a Set request. The string must have only digits and periods and no embedded spaces.

**AsnSelect** – Specifies how data should be formatted in the Set request. The selection is made from the following table:

ASN Select Label	Interpretation
Integer	32-bit signed integer (ASN x02)
TimeTicks	32-bit unsigned integer (ASN x43)
Gauge	32-bit unsigned integer (ASN x42)
Counter	32-bit unsigned integer (ASN x41)
OctetStr	String of 8-bit bytes (ASN x04)
Counter64	64-bit unsigned integer (ASN x46)
OpaqueFloat	Net-SNMP 32-bit floating point (ASN x78)
OpaqueDouble	Net-SNMP 64-bit floating point (ASN x79)
OpaqueU64	Net-SNMP 64-bit unsigned integer (ASN x7B)
OpaqueI64	Net-SNMP 64-bit signed integer (ASN x7A)
OpaqueCounter64	Net-SNMP 64-bit unsigned integer (ASN x76)
BitStr	SNMP bit string (ASN x03)

ObjectID	SNMP object ID (ASN x05)
IpAddress	32-bit IPv4 address (ASN x40)
OctetStrFloat	RFC 6340 32-bit floating point, Octet String (ASN x04, hint FLOAT)
OctetStrDouble	RFC 6340 64-bit floating point, Octet String (ASN x04, hint DOUBLE)

**SendPeriodic** – Set to “N” to disable, or “Y” to enable periodic sending of the Set request at the poll rate given by PollTime.

**PollTime** – Poll time in seconds, can be fractional. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again. The sets the rate at which the remote SNMP device will be updated, provided “sendPeriodic” has been enabled. This poll time will be disregarded if sendPeriodic is not enabled.

**SendDelta** – Set to “N” to disable, or “Y” to enable the “send on delta” feature where Set requests are made based on changes in the local object value (see delta below).

**Delta** – Specifies the margin by which the local object value should change before sending another Set request to send the new value to a remote SNMP device. Once the changed value has been sent, the new local value is retained for future comparison in determining subsequent additional change. The delta value is disregarded if sendDelta is not enabled.

**SendMaxQuiet** – Set to “N” to disable or “Y” to enable the MaxQuietTime feature. If disabled, the MaxQuietTime will be disregarded.

**MaxQuietTime** – Time in seconds, can be fractional. If a Set request has not been made either as a result of poll timing or value changing by delta within this time period, then a Set request will be made anyway. This specifies the maximum amount of time that should expire without any Set request being made for any reason.

**MinQuietTime** – Time in seconds, can be fractional. This specifies the minimum amount of time that should elapse between sending of Set requests for this write map. The minimum quiet time has the effect of throttling network traffic.

**IndexObj** – Optional, allows for selectively enabling this read operation. If an index object (local object number) is given, and its value matches the IndexVal value, then this read operation will take place. If an IndexObj is given but the local object’s value does not match the IndexVal, then this read operation will be skipped.

**IndexVal** – Optional, used in conjunction with IndexObj (see note above).

## SNMP (client) WALKRULES Section

**Device** – Identifies which device from the device list this rule applies to, i.e., determines which IP address to try to establish a connection with.

**OID** – A dot field character string specifying the starting OID for the table walk. The table is walked using GetNext requests, therefore the OID must specify an OID preceding the first OID that one wishes to read.

The table OID is different than OIDs for read and write maps. The table OID includes optional wildcard fields when the walk method is Index or Mask. For example, to walk the alarm table of a UPS using RFC 1628, the OID would be 1.3.6.1.2.1.33.1.6.2.1.\*(2)\*. The \*(2) means allow any value in the second to last field but only act on the value when this field is 2. The last asterisk means disregard the last field entirely (which for RFC 1628 increments with each new alarm until reaching some rollover point, then starts back at 1 again).

**Method** – Defines the method of table walk to be used with values as follows:

Method Label	Method
Normal	Normal – see documentation
Sparse	Sparse – see documentation
Wildcard	Wildcard – see documentation
Index	Index – see documentation
Mask	Mask – see documentation

The table walking process can become complicated by the fact that tables are allowed to have missing rows, or rows may have intermittent missing columns. In some applications, rows will be populated only temporarily and then they will disappear. The table walk method identified as "Normal" expects the given column to exist in all rows through the range of rows defined by the starting OID and the count. The remaining methods accommodate the other complications permitted by SNMP. (See additional documentation for full description of methods.)

**Hint** – Provides direction on how to interpret data in the event the value type is Octet String. All ASN data types are interpreted according to their type; however, Octet String will default to being interpreted as a character string unless this hint says otherwise, and the length is exactly 4 or 8 bytes (octets).

Hint Label	Description
None	No special interpretation

Float      Treat 4-byte Octet String as RFC 6340 IEEE 754 32-bit floating point

Double     Treat 8-byte Octet String as RFC 6340 IEEE 754 64-bit floating point

**StartObj** – Specifies the first local object number that may be filled by this table walk, used in conjunction with count (below).

**ObjCount** – Specifies the number of local objects, starting at StartObj, that may be filled by this table walk. Most table walks are going to result in filling multiple objects with data.

**PollTime** – Poll time in seconds, can be fractional. The sets the rate at which the table in the remote SNMP device will be walked. This poll time is not guaranteed to be met. Polling is done in round-robin fashion. In a very busy system, more than this time may expire before the next poll. If less than this time has expired, then the system will wait this amount of time until polling again.

**IndexObj** – Optional, allows for selectively enabling this read operation. If an index object (local object number) is given, and its value matches the IndexVal value, then this read operation will take place. If an IndexObj is given but the local object's value does not match the IndexVal, then this read operation will be skipped.

**IndexVal** – Optional, used in conjunction with IndexObj (see note above).