

BB4-8422 SNMP Agent

Overview

The local data objects within the IoTServer can be read and written as SNMP variables via the SNMP Agent. The local objects cannot be accessed directly - they need to be mapped to one of the branches of the IoTServer's MIB. The IoTServer supports four branches, each having a specific data format for sharing of data. The most universally accepted data format in SNMP is the simple integer. Regardless of the internal data format assigned to a local object, it will be converted to the data type associated with the corresponding MIB branch when an SNMP Get request is processed. Likewise when an SNMP Set is performed, the data will be automatically converted to the native internal data type assigned to the local object.

The branches of the IoTServer MIB are as follows:

- 1.3.6.1.4.1.3815.1.5.1.1 Integer, signed, 32-bit representation of local objects
- 1.3.6.1.4.1.3815.1.5.1.2 Integer, unsigned, 64-bit representation of local objects
- 1.3.6.1.4.1.3815.1.5.1.3 Floating point representation of local objects
- 1.3.6.1.4.1.3815.1.5.1.4 Character string representation of local objects
- 1.3.6.1.4.1.3815.1.5.1.5 Trap send rule parameters included in trap messages
- 1.3.6.1.4.1.3815.1.5.1.6 Trap send data included in trap messages
- 1.3.6.1.4.1.3815.1.5.1.7 Trap OID identifying a trap message

Status

SNMP Agent MIB-Int32 Data

Dashboard / Protocol Status / SNMP Agent Status

MIB-Int32

MIB-Uint64

MIB-Real

MIB-Char

Traps

Devices

Show 10 entries

Search:

Index	Local Object	Present Value
1	12 : Object 12	0
2	13 : Object 13	0
3	14 : Object 14	0
4	15 : Object 15	0
5	16 : Object 16	1100
Index	Local Object	Present Value

Showing 1 to 5 of 5 entries

Previous 1 Next

This status page shows a list of local objects assigned or mapped to the indicated index positions in this branch of this IoTServer's MIB. The present value of the local object is also displayed.

SNMP Agent MIB-Uint64 Data

Dashboard / Protocol Status / SNMP Agent Status

MIB-Int32 **MIB-Uint64** MIB-Real MIB-Char Traps Devices

Show 10 entries

Search:

Index	Local Object	Present Value
1	3 : Object 3	0
Index	Local Object	Present Value

Showing 1 to 1 of 1 entries

Previous 1 Next

This status page shows a list of local objects assigned or mapped to the indicated index positions in this branch of this IoTServer's MIB. The present value of the local object is also displayed.

SNMP Agent MIB-Real Data

Dashboard / Protocol Status / SNMP Agent Status

MIB-Int32 MIB-Uint64 **MIB-Real** MIB-Char Traps Devices

Show 10 entries

Search:

Index	Local Object	Present Value
1	4 : Object 4	0
2	5 : Object 5	0
Index	Local Object	Present Value

Showing 1 to 2 of 2 entries

Previous 1 Next

This status page shows a list of local objects assigned or mapped to the indicated index positions in this branch of this IoTServer's MIB. The present value of the local object is also displayed.

SNMP Agent MIB-Char Data

Dashboard / Protocol Status / SNMP Agent Status

MIB-Int32 MIB-Uint64 MIB-Real MIB-Char Traps Devices

Show 10 entries

Search:

Index	Local Object	Present Value
1	6 : Object 6	0
2	7 : Object 7	Test char string
Index	Local Object	Present Value

Showing 1 to 2 of 2 entries

Previous 1 Next

This status page shows a list of local objects assigned or mapped to the indicated index positions in this branch of this IoTServer's MIB. The present value of the local object is also displayed.

SNMP Agent Trap Status

Dashboard / Protocol Status / SNMP Agent Status

MIB-Int32 MIB-Uint64 MIB-Real MIB-Char Traps Devices

Show 10 entries

Search:

Rule number	Present state	Seconds since last send	Times sent	Error Code	Error Message
1	0	4,409.68	0	SNMP error 0	Success
Rule number	Present state	Seconds since last send	Times sent	Error Code	Error Message

Showing 1 to 1 of 1 entries

Previous 1 Next

 Clear Counts

The SNMP Agent Trap Status page shows a list of all defined traps that could be sent along with a tally of how many times sent and when last sent. If errors were encountered in attempting to send a trap, the error indication is also displayed.

Counts may be cleared by clicking the Clear button. Clearing counts simply allows you to monitor the occurrence of new trap messages.

SNMP Agent Device Status

Dashboard / Protocol Status / SNMP Agent Status

MIB-Int32

MIB-Uint64

MIB-Real

MIB-Char

Traps

Devices

Show entries

Search:

Device Number	Peer Name	SNMP Error	SysError
1	192.168.1.109:162	SNMP error 0	Success
Device Number	Peer Name	SNMP Error	SysError

Showing 1 to 1 of 1 entries

Previous **1** Next

 Clear status

The Trap Status page provides diagnostic information on a rule by rule basis. This page shows diagnostic information on a device by device basis.

Errors may be cleared by clicking the Clear button. Clearing errors simply clears present error indications so you can check for the occurrence of new errors.

Configuration

NOTE: Any time the definition of the MIB is changed by adding or removing variables in the MIB branches, it is necessary to restart SNMP to reload the new definition of the MIB. When the MIB is changed as illustrated in any of the "Mapping" sections below, go to the SNMP Agent Config File page and click the Restart SNMP button at the bottom of that page.

SNMP Agent MIB-Int32 Mapping

Dashboard / Protocol Configuration / SNMP Agent

MIB-Int32

MIB-Uint64

MIB-Real

MIB-Char

Traps











Devices

Config File

Snmpd.conf

Show 10 entries

Search:

Index	Local Object	Scale
1  	12 : Object 12	0
2  	13 : Object 13	0
3  	14 : Object 14	0
4  	15 : Object 15	10
5  	16 : Object 16	100
Index	Local Object	Scale

Showing 1 to 5 of 5 entries

Previous 1 Next

Mib Index:

Add/Edit mib(s)

Local objects are assigned to the Int32 MIB branch on this page. As the name implies, objects mapped to this branch will be accessed by SNMP Get/Set as 32-bit signed integer values. If the local object is not an integer, it will be converted to/from integer when the Get/Set is performed.

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

To add or insert a new index, enter an index number and click Add/Edit.

Int32 MIB Branch, Index 1

Assign local object named to index 1.



Scale:

There are only two things to enter on the index edit page for integer: The object that should be mapped to this MIB index, and a scale factor. The only universally recognized method of sharing real data is scaled integer. Therefore, a scale factor is available for this purpose. If left set to zero, it will be interpreted as "no scale" and be mathematically treated as 1.0.

SNMP Agent MIB-Uint64 Mapping

MIB-Int32 **MIB-Uint64** MIB-Real MIB-Char Traps Devices Config File Snmpd.conf

Show entries Search:

Index	Local Object	Scale
1  	3 : Object 3	0
Index	Local Object	Scale

Showing 1 to 1 of 1 entries Previous 1 Next

Mib Index: Add/Edit mib(s)

Local objects are assigned to the Uint64 MIB branch on this page. Objects mapped to this branch will be accessed by SNMP Get/Set as 64-bit unsigned integer values (ASN Counter64). If the local object is not an unsigned integer, it will be converted to/from unsigned integer when the Get/Set is performed.

- Click on the index number or pencil icon to jump to the index editing page for this index.

- Click on the trash can icon to delete this index.

Dashboard / Protocol Configuration / SNMP Agent / Uint64 MIB Branch, Index 1

Uint64 MIB Branch, Index 1

Assign local object named to index 1.

Scale:





There are only two things to enter on the index edit page for unsigned integer: The object that should be mapped to this MIB index, and a scale factor. Scale factor is optional. If left set to zero, it will be interpreted as "no scale" and be mathematically treated as 1.0.

SNMP Agent MIB-Real Mapping

Dashboard / Protocol Configuration / SNMP Agent

MIB-Int32 MIB-Uint64 **MIB-Real** MIB-Char Traps Devices Config File Snmpd.conf

Show entries Search:

Index	Local Object
1  	4 : Object 4
2  	5 : Object 5
Index	Local Object

Showing 1 to 2 of 2 entries Previous 1 Next

Mib Index: Add/Edit mib(s)

Local objects are assigned to the Real MIB branch on this page. Objects mapped to this branch will be accessed by SNMP Get/Set as floating point values. If the local object is not a Real, it will be converted to/from Real when the Get/Set is performed.

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

Dashboard / Protocol Configuration / SNMP Agent / Real MIB Branch, Index 1

Real MIB Branch, Index 1

Assign local object named to index 1.

There is only one thing to enter on the index edit page for Real: The object that should be mapped to this MIB index.

There is no one universally recognized format for sharing Real data via SNMP. The IoTServer supports four options, and these are selected in configuration of the SNMP Agent task via the Task Manager.

Dashboard / System / Task Manager

System Task:

SNMP Agent Configuration

MIB Table Size:

Encoding of Floating Point:

Number of Trap Send Rules:

- ASN Opaque Float
- ASN Opaque Double (default)**
- RFC 6340 Float
- RFC 6340 Double

SNMP Agent MIB-Char Mapping

Dashboard / Protocol Configuration / SNMP Agent

MIB-Int32 MIB-Uint64 MIB-Real MIB-Char Traps Devices Config File Snmpd.conf


Show 10 entries

Search:

Index	Local Object
1  	6 : Object 6
2  	7 : Object 7
Index	Local Object

Showing 1 to 2 of 2 entries

Previous 1 Next

Mib Index:  [Add/Edit mib\(s\)](#)

Local objects are assigned to the Char MIB branch on this page. Objects mapped to this branch will be accessed by SNMP Get/Set as Octet String (UTF-8) values. If the local object is not a character string, it will be converted to/from a character string when the Get/Set is performed.

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

Dashboard / Protocol Configuration / SNMP Agent / Char MIB Branch, Index 1

Char MIB Branch, Index 1

Assign local object  named to index 1.



There is only one thing to enter on the index edit page for Char: The object that should be mapped to this MIB index.

SNMP Agent Trap Send Rules

Dashboard / Protocol Configuration / SNMP Agent

MIB-Int32 MIB-Uint64 MIB-Real MIB-Char **Traps** Devices Config File Snmpd.conf

Show entries Search:

Rule number	Branch	Table index
1  	1	1
Rule number	Branch	Table index

Showing 1 to 1 of 1 entries Previous 1 Next

Add new trap send rules(s) before rule number: Insert trap(s)

The SNMP Agent has the ability to automatically send SNMP Traps (or Informs) when certain criteria is met regarding values found in local objects in the IoTServer. The list of SNMP Trap Send Rules currently defined is found on this page.

NOTE: Trap rules do not directly reference local objects. The local object that is to be the subject of a trap rule must first be assigned a spot in the MIB. The trap rule will then reference the branch and index where that object is assigned. This is required because the trap message must include an OID that identifies the data point, and this can only be done by assigning the object to a place in the MIB so that it now has an OID.

Rule number	Branch	Table index
1  	1	1

- 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 trap send rules(s) before rule number:

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

SNMP Agent Trap Send Rule Edit

SNMP Trap or Inform messages sent by this IoTServer are defined by rules outlined here. Traps may be sent as v1, v2c, or v3, and this selection is made in the SNMP Trap Destination Device configuration. The same trap may be sent to different devices as both v2c and v3 at the same time using the same Trap Send Rule.

Dashboard / Protocol Configuration / SNMP Agent / Trap Rule #1

Trap Send Rule #1

Rule Number – Used as a reference in the rule list for ordering the rules.

Using local object found at:

MIB Branch: Table index:

Branch – Specifies which branch in the MIB that this trap rule applies to, and primarily provides the means to look up the local object number that should be evaluated.

Index – Specifies the index or row number in the given branch for looking up the local object number.

Apply test:

Test type:

Test – Defines the test that should be performed to determine if the trap send rule state is true or false. Trap state is considered “true” when this condition is met

Test label **Test performed on object versus threshold**

GT Object greater than threshold

GE	Object greater than or equal threshold
LT	Object less than threshold
LE	Object is less than or equal threshold
EQ	Object is equal threshold
NE	Object is not equal threshold
DEV	Object deviates from threshold by hysteresis amount
DELTA	Object has changed by threshold amount

For most tests, the object is simply compared to the threshold value. The delta test is a special case. The threshold specifies an amount by which the local object needs to change before the rule test will be flagged as true. However, this “true” state is only temporary. Once the trap is sent, the new object value is now saved for subsequent tests of “changed by”. Every time the local object changes by the threshold amount, a new trap will be sent.

Test type delta with threshold of zero is a special case within the special case. If the test type is delta and the threshold value is zero, then the trap will be sent any time the local object (found by looking it up in the MIB) has been updated or changed by some other action in the system, without any regard for what the actual value of the object is.

Apply to:

Threshold value: 100 or Threshold Object: 0

Hysteresis: 0

Threshold Value (CSV "CondVal") – Provided no threshold object number is given, this becomes the threshold value for test purposes.

Threshold Object (CSV "CondObj")– Overrides the threshold value when given (non-zero), and provides the local object from which a test threshold should be retrieved.

Hysteresis (CSV "Hyst") – Specifies the hysteresis value to be applied in the test process. Hysteresis is used to prevent a flood of trap messages when the object is hovering near the threshold but fluctuating frequently. For ‘greater than’ type tests, once the rule state becomes true, the object value must fall below the threshold by the hysteresis amount before the rule state will return to false. For ‘less than’ type tests, once the rule state becomes true, the object value must rise above the threshold by the hysteresis amount before the rule state will return to false. For example, if the rule threshold is 10, and the test is ‘greater than’, then the rule state will

become true when the object value exceeds 10. If the hysteresis value is 2, then the object value must now fall below 8 before the rule state will return to false.

The hysteresis value takes on a special role when the test type is “deviates by”. The rule state will be true when the difference between object value and threshold exceeds the hysteresis amount in either direction, high or low.

Send Trap on condition TRUE Send Trap on condition FALSE

SendOnTrue – Select (use Y/N in CSV) to enable the sending of a trap when the condition specified by this rule tests “true”.

SendOnFalse – Select (use Y/N in CSV) to enable the sending of a trap when the condition specified by this rule tests “false”.

Sending trap message to device groups:

A B C D E F G H

Device Group – The device group allows selectively sending the same trap to multiple devices. Both the trap send rules and the trap devices have a group association. When the group association of a trap rule matches the groups that the device is a member of, the trap will be sent to that device, and all devices included in the group.

The group selection is made as a bit mask labeled "DevMask" in CSV and XML files. Group "A" is bit 0 or value of 1. Group "B" is bit 1 or value of 2. Group "C" is bit 2 or value of 4, and so on. The mask is the summation of the groups. Only the first 8 bits are used in the web UI for ease of use. Internally, the test for group membership is a simple logical AND of the mask values found in the trap rule and the device configuration.

Send Inform instead of Trap

SendInform – Uncheck box (default) to send the message as an SNMP Trap. Check box to send the message as an SNMP Inform. In the CSV or XML file, use "Y" to send as Inform, "N" to send as trap. If omitted, the default is to send an SNMP Trap.

With these optional qualifications:

Minimum True time: 0 seconds Minimum False time: 0 seconds

OnTime – Specifies a minimum amount of time in seconds that the “true” state must exist before the trap state will be fully regarded as true and “true” trap sent. If the condition tests true, and within this time period returns to false, no “true” trap will be sent.

OffTime – Specifies the minimum amount of time in seconds that the “false” state must exist before the trap state will be fully regarded as false and the “false” trap sent. If the condition tests false, and within this time period returns to true, no “false” trap will be sent.

Include optional message fields:

When True: Value is up	When False: value is down
------------------------	---------------------------

TrueMsg – Provides a user defined message that will be delivered as one of the varbinds in any “true” Trap or Inform message sent.

FalseMsg – Provides a user defined message that will be delivered as one of the varbinds in any “false” Trap or Inform message sent.

Repeat 0	times when true	Repeat 0	times when false
Repeat sending every: 0		seconds	

RepTrue – Used to repeat the trap this number of times when true. Traps are not acknowledged, so this provides a means of repeating the same Trap message to better ensure delivery. Set to -1 to repeat indefinitely while condition tests true.

RepFalse – Used to repeat the trap this number of times when false. Traps are not acknowledged, so this provides a means of repeating the same Trap message to better ensure delivery. Set to -1 to repeat indefinitely while condition tests false.

RepTime – Specifies the amount of time in seconds to wait in between repeated sending of the same Trap message as indicated by the repeat count attributes above.

SNMP Agent Trap Destination Devices

Dashboard / Protocol Configuration / SNMP Agent

MIB-Int32 MIB-Uint64 MIB-Real MIB-Char Traps **Devices** Config File Snmpd.conf

Show entries

Search:

Device Number ↑↓	Peer Name ↑↓	Local Name ↑↓
1 	192.168.1.109:162	Dell
Device Number	Peer Name	Local Name

Showing 1 to 1 of 1 entries

Previous Next

Add/Edit

Select device

The SNMP Trap Destination Devices page is where you set up the list of SNMP managers that this IoTServer will send SNMP Traps or Informs to.

Device Number ↑↓	Peer Name ↑↓	Local Name ↑↓
1 	192.168.1.109:162	Dell

- 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

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 Agent Device Edit

The DEVICES section in the SNMP server (agent) specify which remote SNMP devices traps or informs should be sent to as a result of the trap send rules.

Dashboard / Protocol Configuration / SNMP Agent / SNMP Trap Destination Device #1

Trap Destination Device Number: 1

Number – A number ranging from 1 to device table size, and was historically referenced in read and write maps as the device to which the map applied. However, with the implementation of device mask in the SNMP agent, the mask is what actually determines which device(s) the trap is sent to, and the same trap may be sent to multiple devices with only one trap rule as a result of the mask implementation. This number is therefore simply a row number on the list for database reference.

Local Name: Dell

Name – Simply a reference in the web UI for the user to identify this device.

Peer Name (Domain or IP Address): 192.168.1.109:162

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.

Include in these groups:

A B C D E F G H

Device Group – Select which groups this device is a member of. The device group allows selectively sending the same trap to multiple devices. Both the trap send rules and the trap devices have a group association. When the group association of a trap rule matches the groups that the device is a member of, the trap will be sent to that device, and all devices included in the group.

The group selection is made as a bit mask labeled "DevMask" in CSV and XML files. Group "A" is bit 0 or value of 1. Group "B" is bit 1 or value of 2. Group "C" is bit 2 or value of 4, and

so on. The mask is the summation of the groups. Only the first 8 bits are used in the web UI for ease of use. Internally, the test for group membership is a simple logical AND of the mask values found in the trap rule and the device configuration.

SNMP Version: 3 ▾	SNMP v1/v2c Community: oakpark
-------------------	--------------------------------

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

Community – Is 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]
Engine ID: 0x80000ee7031803731a238	

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:** This string must be a minimum of 8 characters long. If less than 8 characters, authentication is guaranteed to fail.

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:** This string must be a minimum of 8 characters long. If less than 8 characters, decryption is guaranteed to fail.

EngineId - Sets the engine ID that will be sent with the trap message if SNMPv3. (Used only for SNMPv3.)

NOTE: The engine ID will be taken as a literal ASCII string (and probably not work) if it does not begin with “0x”. The recipient of an SNMPv3 trap will generally discard the message if the engine ID does not match its own engine ID. It is necessary to know quite a bit about where you are sending traps with v3.

SNMP Agent Config File

Dashboard / Protocol Configuration / SNMP Agent

MIB-Int32 MIB-Uint64 MIB-Real MIB-Char Traps Devices **Config File** Snmpd.conf

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:

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:	agent2.csv	 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.

Configuration File Load/Save

Configuration file:	snmpdGenerated.conf	 Refresh
 Load Conf file into Engine		 Save Current Config into .conf file

Snmpd.conf Files: Any time the snmpd.conf file is regenerated on the Snmpd.conf page, you will need to come here to transfer that generated file into the SNMP engine. Select the generated .conf file from the drop-down list below, and then click the Load button. You can also retrieve a copy of the snmpd.conf file actually in use by clicking the Save button. The content of the currently in-use snmpd.conf file will be transferred to the file name you have entered.

NOTE: Any time you reload the snmpd.conf file here, you also need to click the Restart SNMP button at the bottom of this page.

Configuration File Management

Available Configuration Files:

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.

Configuration Error Logs

Configuration Error Logs:

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.

SNMP Logging File

Logging enabled:

Select Yes to enable logging, or No to disable. When selecting Yes, provide a log name in the /home/customer/logs/ directory. All accesses to the MIB by external managers are logged here. It is recommended that you enable logging only temporarily for diagnostic purposes to avoid eventually running out of file space.

The log files can be viewed on the System -> Logs page.



The SNMP Agent task 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.

The SNMP Agent task suspended via the Suspend button is the API task that provides the interface between the web UI and the internal task management. The SNMP Engine itself is another process. Any time the MIB configuration is altered, it is necessary to restart SNMP (snmpd service). Click Restart SNMP to reload SNMP with the new definition of the MIB.

SNMP Agent Snmpd.conf File



The SNMP engine that responds to external SNMP manager's Get and Set requests requires a configuration file named snmpd.conf to direct its functionality, primarily in terms of authorization. This page allows automated generation of that file, but with the option to manually edit it.

The SNMP engine in this IoTServer is the widely used open source Net-SNMP package. If you are concerned about manually editing the snmpd.conf file, simple search the Internet for Net-SNMP documentation - there is much to be found.

Agent settings

The port on which SNMP listens for Get/Set requests defaults to the standard port 161. You may change it here if you wish.

System Group Parameters

All SNMP devices have a set of system parameters that are universally available, and these describe the system. Enter your personalized system information here.

SNMP versions 1 and 2

Authorization for access in SNMPv1 and SNMPv2 is very simply. You only need to match the community names which are treated sort of like a password. If using SNMP v1 or v2 (v2c), enter your community strings here.

[Generate Config](#) 

Note: Generating the configuration here will not rewrite SNMP Agent Configuration. It will generate a preview of the configuration file. User authentication information is taken from user settings found on the Users page. After generating the preview, you can then save the preview as a file below. Then go to the Config File page to import that file into the actual SNMP agent.

Click the Generate Config button to auto-generate an snmpd.conf file. Parameters included in the file will be taken from two places: (1) The entries showing above on this page. (2) User settings from the System -> Users page (if SNMPv3).

SNMPv3 enforces access only by known users. These users may be a person or may be another machine, but must be defined as a "user" either way. To permit SNMPv3 access, go to the System -> Users page and create a "user" for each person or machine that will access the MIB in this IoTServer. Once the users are created, and they have been selected for SNMPv3 access, they will be automatically included in the snmpd.conf file generated here.

Preview of generated SNMP agent configuration

```
agentaddress 161

rocommunity public
rwcommunity oakpark

#Users for agent defined here

engineIDType 3 engineIDNic eth0

# System Group
sysDescr Control Solutions IoTServer
sysObjectID .1.3.6.1.4.1.3815.2.5.1.1

# Override default values from 'configure' - makes these objects read-only
```

Save config to file:

Once the interim snmpd.conf file is generated, it may be viewed and optionally edited here. The file content displayed here is not actually saved to a file until you click the Save button at the bottom. When Save is clicked, the file name given will be created or overwritten with the content currently displayed.

There are two more steps required to complete the reconfiguration of SNMP. Go to the SNMP Agent Config File page, select the newly created interim snmpd.conf file, and click the Load button (as discussed under "SNMP Agent Config File"). Finally, after loading the new snmpd.conf file, click Restart SNMP at the bottom of the Config File page.

Dashboard / Protocol Configuration / SNMP Agent

The snmpdGenerated.conf file has been saved to your configs directory. You can now load it into snmp using the 'Config File' tab.

When the snmpd.conf file content is saved to an actual file as a result of clicking the Save button, you should see the confirmation illustrated here.

SNMP Agent XML Files

Example XML File

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

<configuration>

<mib_vars_int32>
<var index="1" object="1"/>
<var index="2" object="2"/>
<var index="3" object="3"/>
<var index="4" object="4" scale="10"/>
<var index="5" object="5" scale="100"/>
</mib_vars_int32>

<mib_vars_uint64>
<var index="1" object="3"/>
</mib_vars_uint64>

<mib_vars_double>
<var index="1" object="4"/>
<var index="2" object="5"/>
</mib_vars_double>

<mib_vars_charstr>
<var index="1" object="6"/>
<var index="2" object="7"/>
</mib_vars_charstr>

<trap_devices>
<dev id="1" version="3" secLevel="3" peername="192.168.1.109:162" name="Dell"
devMask="1" username="jim" authType="MD5" authPhrase="jimsAuthPhrase"
privType="DES" privPhrase="jimsPrivPhrase" engineId="0x80000ee7031803731a2386"/>
</trap_devices>
```

```

<trap_rules>
<rule branch="1" index="1" test="gt" condval="10" sendOnTrue="1" sendOnFalse="1"
devMask="1" truemsg="Value is up" falsemsg="value is down"/>
</trap_rules>

</configuration>

```

<mib_vars_xxx> section

There will be up to four <mib_vars_xxx> sections in the XML file referring to the branches of the MIB. The `setMibVars` API call specifies the branch by number in the call. The contents of each MIB var entry are as listed below - a simple mapping of object to MIB index.

index="n" – Refers to the row number in the MIB table, ranging from 1 to maximum table size as set in the agent task configuration at startup.

object="n" – Refers to the local object number mapped at this location (branch and index) in the MIB.

scale="n.nn" – Applies to integer branches only (disregarded otherwise), and provides the scale factor by which the local object value is multiplied when responding to an SNMP Get, or by which the incoming value from an SNMP Set is divided before placing it into the local object.

<trap_devices> section

id="n" – A number ranging from 1 to device table size, and was historically referenced in read and write maps as the device to which the map applied. However, with the implementation of device mask in the SNMP agent, the mask is what actually determines which device(s) the trap is sent to, and the same trap may be sent to multiple devices with only one trap rule as a result of the mask implementation.

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.

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

devMask="x" – A 32-bit bit mask that allows sending the same trap to multiple devices. Both the trap send rules and the trap devices have a “device mask” (`devMask`). This effectively creates device groups in a very simple manner. The bit mask found in the trap send rule is logically ANDed with the bit mask in the device table entry. If the result is non-zero, then the trap is sent to this device.

community="xxxx" – Is the community string as defined for SNMP v1 and v2c.

name="xxxx" – Simply a reference in the web UI for the user to identify this device.

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.)

engineId="xxx" - Sets the engine ID that will be sent with the trap message if SNMPv3. (Used only for SNMPv3.)

Note: The engine ID will be taken as a literal ASCII string (and probably not work) if it does not begin with “0x”. The recipient of an SNMPv3 trap will generally discard it if the engine ID does not match its own engine ID. It is necessary to know quite a bit about where you are sending traps with v3.

<trap_rules> section

branch="n" – Specifies which branch in the MIB that this trap rule applies to, and primarily provides the means to look up the local object number that should be evaluated.

index="n" – Specifies the index or row number in the given branch for looking up the local object number.

condval="n.nn" – Provided no conditional object number is given, this becomes the threshold value for test purposes.

condobj="n" – Overrides the conditional value when given (non-zero), and provides the local object from which a test threshold should be retrieved.

test="xxx" – Defines the test that should be performed to determine if the trap send rule state is true or false.

Trap state is considered “true” when this condition is met

XML Label Test Code Rest performed on object versus threshold

“none”	0	No trap sent
“gt”	1	Object greater than threshold
“ge”	2	Object greater than or equal to threshold
“lt”	3	Object less than threshold
“le”	4	Object is less than or equal to threshold
“eq”	5	Object is equal to threshold
“ne”	6	Object is not equal to threshold
“dev”	7	Object deviates from threshold by hysteresis amount
“delta”	8	Object has changed by threshold amount

Threshold is either the fixed value given by “condval”, or the value found in the object given as “condobj”.

For most tests, the object is simply compared to the threshold value. The delta test is a special case. The threshold specifies an amount by which the local object needs to change before the rule test will be flagged as true. However, this “true” state is only temporary. Once the trap is sent, the new object value is now saved for subsequent tests of “changed by”. Every time the local object changes by the threshold amount, a new trap will be sent.

Test type delta with threshold of zero is a special case within the special case. If the test type is delta and the threshold value is zero, then the trap will be sent any time the local object (found by looking it up in the MIB) has been updated or changed by some other action in the system, without any regard for what the actual value of the object is.

hyst="n.nn" – Specifies the hysteresis value to be applied in the test process. Hysteresis is used to prevent a flood of trap messages when the object is hovering near the threshold but fluctuating frequently. For ‘greater than’ type tests, once the rule state becomes true, the object value must fall below the threshold by the hysteresis amount before the rule state will return to false. For ‘less than’ type tests, once the rule state becomes true, the object value must rise above the threshold by the hysteresis amount before the rule state will return to false. For example, if the rule threshold is 10, and the test is ‘greater than’, then the rule state will become true when the object value exceeds 10. If the hysteresis value is 2, then the object value must now fall below 8 before the rule state will return to false.

The hysteresis value takes on a special role when the test type is “deviates by”. The rule state will be true when the difference between object value and threshold exceeds the hysteresis amount in either direction, high or low.

ontime="n.nn" – Specifies a minimum amount of time that the “true” state must exist before the trap state will be fully regarded as true and “true” trap sent. If the condition tests true, and within this time period returns to false, no “true” trap will be sent.

offtime="n.nn" – Specifies the minimum amount of time that the “false” state must exist before the trap state will be fully regarded as false and the “false” trap sent. If the condition tests false, and within this time period returns to true, no “false” trap will be sent.

sendOnTrue="x" – Expects a value of 0 to disable, or 1 to enable the sending of a trap when the condition specified by this rule tests “true”.

sendOnFalse="x" – Expects a value of 0 to disable, or 1 to enable the sending of a trap when the condition specified by this rule tests “false”.

sendInform="x" – Expects a value of 0 to send the message as an SNMP Trap, or value of 1 to send the message as an SNMP Inform.

devMask="x" – A 32-bit bit mask that allows sending the same trap to multiple devices. Both the trap send rules and the trap devices have a “device mask” (devMask). This effectively creates device groups in a very simple manner. The bit mask found in the trap send rule is logically ANDed with the bit mask in the device table entry. If the result is non-zero, then the trap is sent to this device.

repcounttrue="n" – Used to repeat the trap this number of times when true. Traps are not acknowledged, so this provides a means of repeating the same Trap message to better ensure delivery. Set to -1 to repeat indefinitely while condition tests true.

repcountfalse="n" – Used to repeat the trap this number of times when false. Traps are not acknowledged, so this provides a means of repeating the same Trap message to better ensure delivery. Set to -1 to repeat indefinitely while condition tests false.

reptime="n.nn" – Specifies the amount of time to wait in between repeated sending of the same Trap message as indicated by the repcount attributes above.

truemsg="xxxx" – Provides a user defined message that will be delivered as one of the varbinds in any “true” Trap or Inform message sent.

falsemsg="xxxx" – Provides a user defined message that will be delivered as one of the varbinds in any “false” Trap or Inform message sent.

SNMP Agent 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
- TRAPSENDRULES

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 (csiSnmpClient, csiSnmpAgent, csiSnmpTrapRecv). A fourth application, csiTrapHandler, is also associated with csiSnmpAgent.

SNMP Agent (Server) Example

```

BEGIN,SNMP,MIBVARS
BRANCH,INDEX,OBJECT,SCALE
1,1,1,0.000000
1,2,2,0.000000
1,3,3,0.000000
1,4,4,10.000000
1,5,5,100.000000
2,1,3,0.000000
3,1,4,10.000000
3,2,5,100.000000
4,1,6,0.000000
4,2,7,0.000000
END
BEGIN,SNMP,DEVICES
NUMBER,PEERNAME,VERSION,COMMUNITY,DEVMASK,NAME
1,192.168.1.109,2,special,0001,Dell
END
BEGIN,SNMP,TRAPSENDRULES
BRANCH,INDEX,TEST,SENDONTRUE,SENDONFALSE,CONDOBJ,CONDVAL,HYST,DEV
VMASK,REPTRUE,REPFALSE,REPTIME,ONTIME,OFFTIME,TRUEMSG,FALSEMSG
1,1,GT,Y,Y,0,10.000000,0.000000,0001,0,0,0.000000,0.000000,0.000000,Value is up,value is
down
END

```


SNMP (agent/server) MIBVARS Section

Branch – Specifies which of the 4 data branches of the MIB this object should be made available in. Data will be automatically converted as necessary. Object type does not matter. Regardless of the local object's native data type, the data will be presented to SNMP according to the format for that branch.

Branch number	Data format
1	Signed 32-bit integer (INTEGER)
2	Unsigned 64-bit integer (COUNTER64)
3	32-bit or 64-bit floating point (see note)
4	Octet String

Note: The floating point branch (3) will default to OPAQUE DOUBLE (Net-SNMP type), but may be configured to be 32-bit or 64-bit Opaque Float or Double, or the 32-bit or 64-bit version of RFC 6340 Octet String interpretation. This selection is a configuration property of the SNMP agent task.

Index – Refers to the row number in the MIB table, ranging from 1 to maximum table size as set in the agent task configuration at startup.

Object – Refers to the local object number mapped at this location (branch and index) in the MIB.

Scale – Applies to integer branches only (disregarded otherwise), and provides the scale factor by which the local object value is multiplied when responding to an SNMP Get, or by which the incoming value from an SNMP Set is divided before placing it into the local object.

SNMP (agent/server) DEVICES Section

The DEVICES section in the SNMP server (agent) specify which remote SNMP devices traps or informs should be sent to as a result of the trap send rules.

Number – A number ranging from 1 to device table size, and was historically referenced in read and write maps as the device to which the map applied. However, with the implementation of device mask in the SNMP agent, the mask is what actually determines which device(s) the trap is sent to, and the same trap may be sent to multiple devices with only one trap rule as a result of the mask implementation. This number is therefore simply a row number on the list for database reference.

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 trap, which in turn determines certain aspects of how the trap message is formatted. Version may be 1, 2, or 3 where 2 really means v2c.

Community – Is the community string as defined for SNMP v1 and v2c.

DevMask – A 32-bit bit mask that allows sending the same trap to multiple devices. Both the trap send rules and the trap devices have a “device mask” (DevMask). This effectively creates device groups in a very simple manner. The bit mask found in the trap send rule is logically ANDed with the bit mask in the device table entry. If the result is non-zero, then the trap is sent to this device.

Name – Simply a reference in the web UI for the user to identify this device.

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.)

EngineId - Sets the engine ID that will be sent with the trap message if SNMPv3. (Used only for SNMPv3.)

NOTE: The engine ID will be taken as a literal ASCII string (and probably not work) if it does not begin with “0x”. The recipient of an SNMPv3 trap will generally discard it if the engine ID does not match its own engine ID. It is necessary to know quite a bit about where you are sending traps with v3.

SNMP (agent/server) TRAPSENDERULES Section

Branch – Specifies which branch in the MIB that this trap rule applies to, and primarily provides the means to look up the local object number that should be evaluated.

Index – Specifies the index or row number in the given branch for looking up the local object number.

Test – Defines the test that should be performed to determine if the trap send rule state is true or false. Trap state is considered “true” when this condition is met

Test label Test performed on object versus threshold

GT	Object greater than threshold
GE	Object greater than or equal threshold
LT	Object less than threshold
LE	Object is less than or equal threshold
EQ	Object is equal threshold
NE	Object is not equal threshold
DEV	Object deviates from threshold by hysteresis amount
DELTA	Object has changed by threshold amount

SendOnTrue – Enter “N” to disable, or “Y” to enable the sending of a trap when the condition specified by this rule tests “true”.

SendOnFalse – Enter “N” to disable, or “Y” to enable the sending of a trap when the condition specified by this rule tests “false”.

SendInform – Enter “N” to send the message as an SNMP Trap, or “Y” to send the message as an SNMP Inform.

CondObj – Overrides the conditional value when given (non-zero), and provides the local object from which a test threshold should be retrieved.

CondVal – Provided no conditional object number is given, this becomes the threshold value for test purposes.

Hyst – Specifies the hysteresis value to be applied in the test process (see API notes).

DevMask – A 32-bit bit mask that allows sending the same trap to multiple devices. Both the trap send rules and the trap devices have a “device mask” (devMask). This effectively creates device groups in a very simple manner. The bit mask found in the trap send rule is logically ANDed with the bit mask in the device table entry. If the result is non-zero, then the trap is sent to this device.

RepTrue – Used to repeat the trap this number of times when true. Traps are not acknowledged, so this provides a means of repeating the same Trap message to better ensure delivery. Set to -1 to repeat indefinitely while condition tests true.

RepFalse – Used to repeat the trap this number of times when false. Traps are not acknowledged, so this provides a means of repeating the same Trap message to better ensure delivery. Set to -1 to repeat indefinitely while condition tests false.

RepTime – Specifies the amount of time in seconds to wait in between repeated sending of the same Trap message as indicated by the repeat count attributes above.

OnTime – Specifies a minimum amount of time in seconds that the “true” state must exist before the trap state will be fully regarded as true and “true” trap sent. If the condition tests true, and within this time period returns to false, no “true” trap will be sent.

OffTime – Specifies the minimum amount of time in seconds that the “false” state must exist before the trap state will be fully regarded as false and the “false” trap sent. If the condition tests false, and within this time period returns to true, no “false” trap will be sent.

TrueMsg – Provides a user defined message that will be delivered as one of the varbinds in any “true” Trap or Inform message sent.

FalseMsg – Provides a user defined message that will be delivered as one of the varbinds in any “false” Trap or Inform message sent.