



CELLWATCH 5

Integrated SNMP Agent

Application Note

Document revision:	v3.0
Cellwatch iBMU MIB:	v1.1.0
Cellwatch version:	v5.1.0



Contents

Introduction	2
Quick Reference	2
Summary of Data Provided	3
Localization	3
Configuration	4
Firewall Access	4
Read and Write Communities	4
Configuring Trap Recipients	4
INI File Settings	4
Appendix A – Troubleshooting	6
Logging	6
Problem: SNMP Agent was unable to connect to port 161	6
Appendix B: The iBMU MIB	9
Basic System Information	9
Read/Write VARS	10
Battery and String Tables	12
monBatteryTable (1024)	13
monStringTable (1408)	14
monJarTable (1792)	17
monAlarmHistoryTable (1803)	18
Traps	20
Trap Variables	24

Introduction

Cellwatch 5 onwards includes SNMP functionality built-in (see the **BMS Interface** menu in Cellwatch).

It supports SNMP v1 features with an SMI1 MIB (found in the **ProgramData** folder). A remote network management system (NMS) may query it and subscribe to traps.

The Cellwatch iBMU **MIB** for the provides a complete description of all data provided and the traps sent. It is summarized in this document in the [MIB section](#) below.

Quick Reference

After installing the Cellwatch SNMP Agent, the iBMU MIB can be found in the following location:

```

MIB Directory:
c:\ProgramData\Cellwatch\

MIB file name:
NDSL-BATTERY-MONITOR-SMI1.mib

Default read community: CELLWATCH
Default write community: DEAFCAT
Trap read community: public (cannot be changed)
Listening port: UDP 161
    
```



Summary of Data Provided

The following information can be requested from the SNMP Agent:

- **Basic system information:** Cellwatch machine name, system status, number of batteries, etc.
- **Battery table:** A listing of all batteries, including name status, strings, and average measurements.
- **String table:** A listing of all strings in the Cellwatch system, including the status of each string, and average measurements.
- **Jar table:** A listing of all the jars in the Cellwatch system, including their most recent measurements, alarm thresholds, and block information.
- **Alarm History table:** A listing of the most recent Cellwatch system alarms.

Additionally, the SNMP Agent will broadcast SNMP v1 traps to UDP recipients on the network when the following Cellwatch events occur:

- A user enables / disables Cellwatch scanning
- Voltage alarm starts (high / low)
- Discharge alarm start (5 seconds after current alarm)
- Discharge alarm stop
- Ohmic value alarm (high / low)
- Temperature alarm starts (delayed temperature)
- Temperature alarm stops
- All alarms have ended on a battery
- The user changes any alarm threshold (system, battery, string or jar level)
- Ohmic value scan has ended
- Voltage scan has ended
- Test traps send by user via SNMP Agent

Please refer to the Cellwatch User Guide for details on the conditions that trigger the above alarms.

Localization

Your Cellwatch system is capable of operating in English, French, Chinese, and Korean. The SNMP agent will work with each of these languages, however all alerts transmitted and the MIB are only available in English.

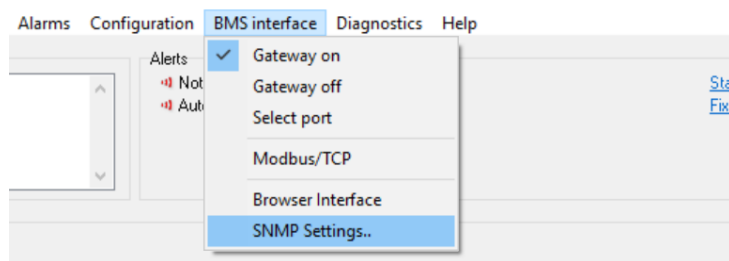


Configuration

To integrate SNMP into your building management system follow these steps:

1. Ensure UDP port 161 is allowed in the BMU firewall (you should be prompted automatically).
2. Configure your READ and WRITE communities (in Cellwatch and your NMS)
3. Add the host or IPs of your trap recipients (maximum of 20 recipients allowed)

Nearly all configuration can be performed via the GUI, under the **BMS Interface** menu:



Firewall Access

When the Cellwatch starts with SNMP enabled it will automatically start listening on UDP port 161. If you see a prompt from the Windows Firewall, accept all requests for permission to communicate using the network.

Read and Write Communities

In SNMP v1 basic security is provided through plain text passwords referred to as **communities**. There is a separate **community** phrase to protect read and write operations.

The default **read** community is **CELLWATCH** (case sensitive) and the default **write** community is DEAFCAT.

If your NMS is not programmed with the correct community your read or write operation will simply timeout (there will be no response from Cellwatch).

To change the READ or WRITE communities, use the settings dialog accessible in the main Cellwatch GUI under **BMS Settings > SNMP Settings** (see figure above).

Configuring Trap Recipients

Cellwatch sends all traps with the read community of **public** - this cannot be changed.

Use the **SNMP Settings** window to configure the trap recipients. Use the **Send Test Trap** button to send a test event (with specific ID 99) to all listed recipients in the GUI.

INI File Settings



The **SNMP** section of the Cellwatch INI file contain the SNMP preferences. The **SNMP Traps** section will contain the trap recipient IPs or hostnames.

Except where marked, these can all be altered via the Cellwatch GUI.

Field name	Default	Meaning
Enabled	False	Set true in order to listen for queries and emit traps. False to disable all SNMP functionality.
Write enabled	False	Set true to allow users to person SNMP SET command on certain registers (see MIB section on read/write vars)
Read community	CELLWATCH	Configured in NMS to allow reading of data. If incorrect Cellwatch will log error and not reply to request.
Write community	DEAFCAT	As above, but for writing (using the SET command). If incorrect Cellwatch will log error and not reply to request.
Alarm limit	20	<p>Not available in the GUI.</p> <p>The number of alarms (starting with most recent) that will be returned in the recent alarms table.</p> <p>Set to -1 to return all the alarms in the database, starting with the most recent.</p>

Trap recipients

This section of the INI file lists all trap recipients (up to a total of 20) in the following format:

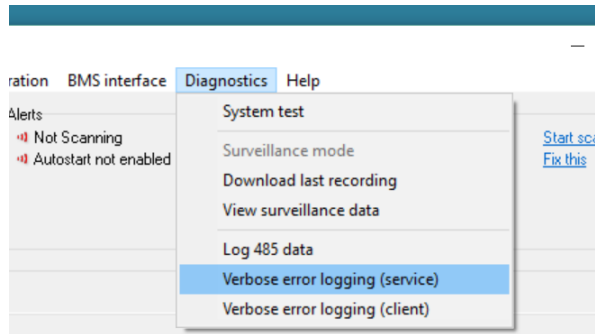
```
[SNMP Traps]
Trap IP 1=192.168.1.1
Trap IP 2=somehost.local
```



Appendix A – Troubleshooting

Logging

When troubleshooting SNMP, it may be useful to enable **Verbose logging** in Cellwatch. To do this, CTRL+click on the **Diagnostics** menu and turn on service side verbose logging:



NOTE: Be sure to disable verbose logging when you've completed your troubleshooting as it will generate large Cellwatch log files.

Problem: SNMP Agent was unable to connect to port 161

Symptom: you cannot query SNMP agent and receive no traps OR see the following line in the Cellwatch log file:

```
15:20:15.999 [error] SNMP agent cannot listen on port: 161 (in use)
```

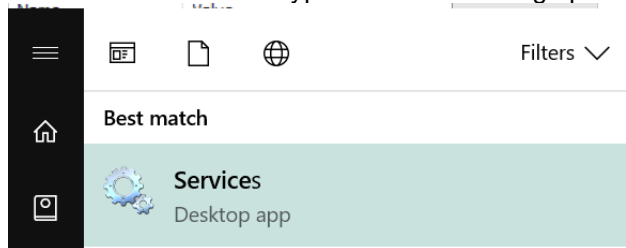
Likely cause: Another piece of software is bound to UDP port 161. This can often be the Windows SNMP service which is enabled on Windows Server editions but not on consumer (desktop) editions.

Remediation:

1. Use a tool like TCPView (available from Microsoft) to find which software is occupying the port.
2. Close that process (if it is a Windows service then also set its startup type to **Disabled**).
3. Open and **Save** the SNMP Settings in Cellwatch (this forces SNMP to try to listen again).

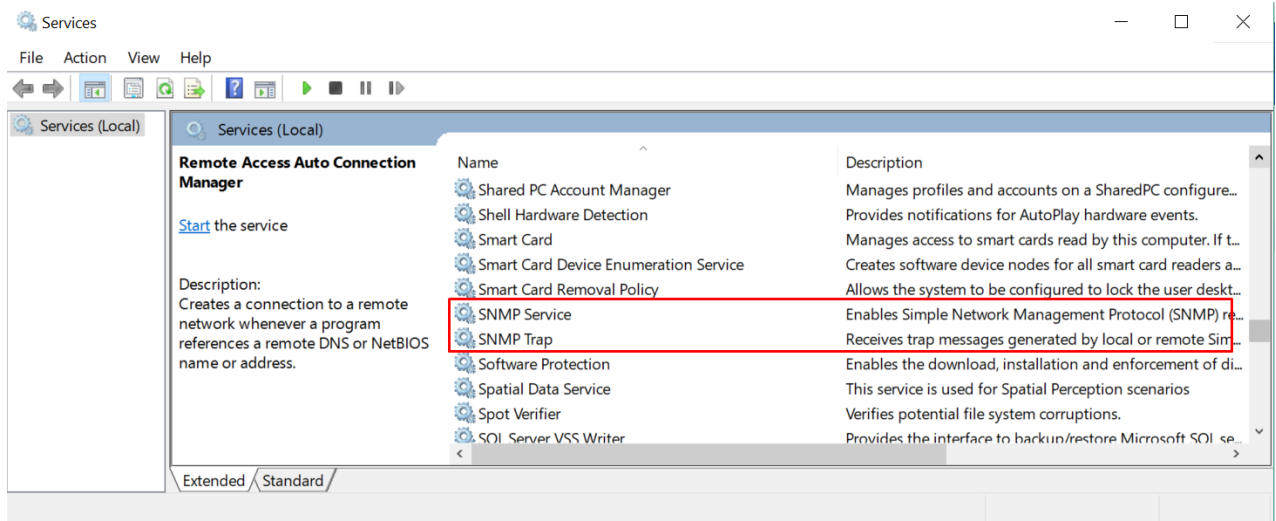
Alternately, the Windows SNMP service may be disabled by following the steps below:

1. Click the Start menu and type **Services** to bring up the service control tab.

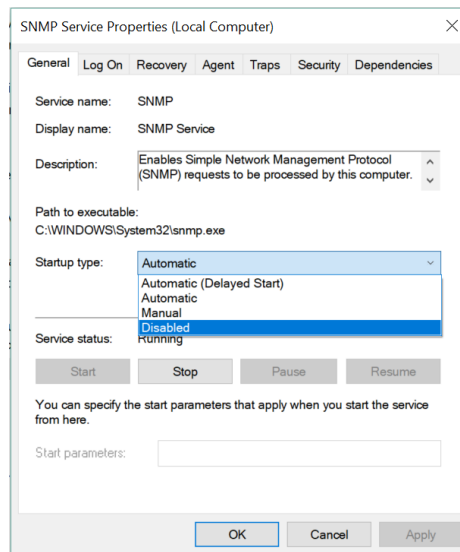




2. Double click **Services** and scroll down to find the two entries: *SNMP Service* and *SNMP Trap Service*.



3. For each service:
 - a. Right click it and select **Stop** (if available). Check the status column afterward to be sure it does not read started.
 - b. Right click the service and click Properties. Set the startup type to disabled, as shown below:



4. Close the Services window.
5. Open and click **Save Changes** in the SNMP Settings Window in Cellwatch. Check the log file for the following line:

15:31:25.208 [notice] SNMP server started. Bound to port: 161



If you are still experiencing problems after following the steps above, please contact your local system administrator for assistance or call Cellwatch Technical Support.

NDSL Support
+1 919 790 7877 x 1
www.cellwatch.com/support/



Appendix B: The iBMU MIB

The iBMU MIB is based on the SMI v1 specification. All objects in this document should be assumed to have the OID prefix 1.3.6.1.4.1.NDSL (11504).BMU (1) in which the NDSL enterprise identifier is 11504.

Within the NDSL node the iBMU has identifier number 1, which is then split into the following two nodes:

VARs (1)

Any object that can be queried by an SNMP manager (both individual objects and tables).

ALERTS (2)

Trap definitions and the variable bindings referenced by traps (to provide supporting information).

Basic System Information

The identifier column contains the object offset relative to the start of the **vars** section in the iBMU MIB, and so the following prefix should be added to get the complete OID:

1.3.6.1.4.1.11504.1.1

(1.3.6.1.4.1.NDSL.BMU.VARS)

Identifier	Name	Data Type	Description
1	monSysName	DisplayString	Unique battery ID
2	monSysScanning	INTEGER	0 = Cellwatch not scanning 1 = Cellwatch scanning
3	monSysStatus	INTEGER	Please see description of status register in the battery and string table section (below).
4	monSysBatteryCount	INTEGER	Number of batteries.
5	monSysStringCount	INTEGER	Total number of strings in this system.
6	monSysAvgBatteryCurrent	INTEGER	Avg. battery current
7	monSysAvgBatteryVoltage	INTEGER	Avg. battery voltage
8	monSysAvgBatteryTemp	INTEGER	Avg. battery temperature
9	monSysMinBatteryCurrent	INTEGER	Min. battery current
10	monSysMinBatteryVoltage	INTEGER	Min. battery voltage
11	monSysMinBatteryTemp	INTEGER	Min. battery temperature
12	monSysMaxBatteryCurrent	INTEGER	Max. battery current
13	monSysMaxBatteryVoltage	INTEGER	Max. battery voltage
14	monSysMaxBatteryTemp	INTEGER	Max. battery temperature
15	monSysOhmicDelay	INTEGER	Minutes into day to delay ohmic scan
16	monSysOhmicScanActive	BOOLEAN	Indicates if Ohmic Scan is active
17	monSysVoltageScanActive	BOOLEAN	Indicates if Voltage Scan is active
18	monSysVoltageScanInt	INTEGER	Indicates the scan mode of Cellwatch for voltage scans (continuously, hourly, every 6 hours)



19	monSysOhmicScanInt	INTEGER	Indicates the scan mode of Cellwatch for ohmic scans (once or twice daily)
----	--------------------	---------	----------------------------------------------------------------------------

Read/Write VARS

The following vars can be both read and written to. Additional information for these variables can be found in the MIB. Modifying these vars will change how the Cellwatch system functions.

Cellwatch 4 introduced writable Modbus registers that can be used to remotely control ohmic value and voltage scans. These registers can be written, if the following entry exists in the Cellwatch INI file:

```
[Modbus]
WriteEnabled=1
```

If the entry is not present or set to zero, the registers cannot be set.

Note: All set requests to the SNMP agent must include only one OID; the agent will reject set requests that contain multiple OIDs.

Object Name	Description
monSysOhmicDelay	<p>The number of minutes into the day that an ohmic value scan occurs.</p> <p>The delay is added to 12:00 AM (e.g., if the delay is set to 1, an ohmic value scan will run at 12:01 AM. If the delay is 60, an ohmic value scan will run at 1:00 AM. If the delay is 600, the ohmic value scan will occur at 10:00 AM. This is often changed so that ohmic alarms occur later in the day.</p>
monSysOhmicScanActive	<p>A value of true(1) indicates that an ohmic value scan is active. false(2) indicates that an ohmic value scan is not active. An ohmic value scan can be started by writing the value true(1) to this object. An ohmic value scan can be cancelled by writing the value false(2) to this object.</p> <p>Cell ohmic data is updated once a completed system ohmic scan has finished. If an ohmic scan is cancelled before the scan has completed the updated data will not be made available, instead the old ohmic data will be available.</p> <p>A general error will be returned if a value of false(2) is written to this object when an ohmic value scan is not active.</p>
monSysVoltageScanActive	<p>A value of true(1) indicates that a voltage scan is active. false(2) indicates that no voltage scan is active. Voltage scans occur so rapidly that this object will usually return a value of false(2) when read unless the system is in constant voltage scanning mode.</p>



	<p>A voltage scan can be started by writing the value true(1) to this object.</p> <p>For systems with constant voltage scanning enabled, this value will always remain as 1 and the user will not be able to initiate a new ohmic scan.</p> <p>A general error will be returned if a value of false(2) is written to this object.</p>
monSysVoltageScanInt	<p>The number of hours between voltage scans. A value of continuously(1) indicates that the system is operating in constant voltage scan mode.</p> <p>everyHour(2) indicates that the system will perform a voltage scan every hour. every6Hours(3) indicates that the system will perform a voltage scan every 6 hours.</p> <p>The recommended voltage scan interval is continuously (1).</p>
monSysOhmicScanInt	<p>The number of hours between ohmic value scans. A value of every12Hours(1) indicates that the system will perform an ohmic value scan every 12 hours. every24Hours(2) indicates that the system will perform an ohmic value scan every 24 hours.</p> <p>The recommended ohmic scan interval is every24Hours(2).</p> <p>Note: Voltage scans will not occur during an ohmic scan.</p>



Battery and String Tables

The battery and string tables show one battery (or string) for each row, with a column for each piece of data provided. In a table layout, the first column is always an index and in the case of the battery and string tables, this uniquely identifies this battery or string within the Cellwatch system.

The second special column is the *status* column (this is labeled **monBatteryStatus** and **monStringStatus** in the battery and string tables, respectively). Each bit in this byte value should be interpreted as follows:

Bit#

0 (LSB) – Hardware communications error.

1 - General Fault - Cellwatch is running, but not scanning. (System only)

2 – Temperature alarm

3 – Current alarm

4 – Ohmic alarm

5 - Voltage alarm

6 – System alarm – Set if any other bit is set

7 – Thermal runaway condition present

8 – A string has been disconnected (must be reset at breaker and in Cellwatch)

9 – String voltage alarm

...

15 (MSB) – Unused



monBatteryTable (1024)

Each row in the battery table corresponds to a Cellwatch battery. Batteries may be multi-string configurations connected to either a single UPS or multiple UPS systems configured in parallel. Due to the paralleled nature of this configuration, they are often referred to as a single battery.

To request a particular column, prefix the column number in the table below by the OID of the first row of the table:

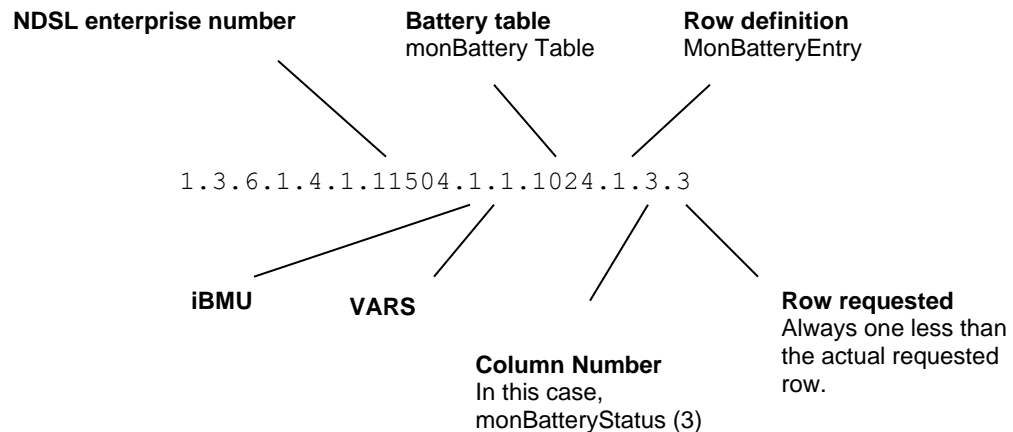
```
1.3.6.1.4.1.11504.1.1.1024.1
(1.3.6.1.4.1.NDSL.BMU.VARS.monBatteryTable.MonBatteryEntry)
```

For example, to request the third column (monBatteryStatus) of the first battery in this system, use this OID in the GET NEXT request:

```
1.3.6.1.4.1.11504.1.1.1024.1.3
```

To read subsequent battery rows (i.e. those other than the first), pass the row identifier (battery number) of the previous row in the table.

For example, to request the third column (monBatteryStatus) of the 4th battery in the system, the OID would appear as follows in the GET NEXT request used to traverse a table.



Identifier	Name	Data Type	Description
1	monBatteryIndex	INTEGER	Unique battery ID
2	monBatteryName	DisplayString	The battery name.
3	monBatteryStatus	INTEGER	Battery status – please see the start of the Battery and Strings Table section for details on this register.
4	monStringCount	INTEGER	Number of strings
5	monStringStartIndex	INTEGER	The index of the first string attached to this battery.
6	monBatteryCurrent	INTEGER	Battery current
7	monBatteryVoltage	INTEGER	Battery voltage.



8	monStringAvgCurrent	INTEGER	Avg. string current in this battery
9	monStringAvgVoltage	INTEGER	Avg. string voltage in this battery.
10	monStringAvgTemp	INTEGER	Avg. string temperature in this battery
11	monStringMinCurrent	INTEGER	Min. string current in this battery.
12	monStringMinVoltage	INTEGER	Min. string voltage in this battery.
13	monStringMinTemp	INTEGER	Min. string temperature in this battery.
14	monStringMaxCurrent	INTEGER	Max. string current in this battery.
15	monStringMaxVoltage	INTEGER	Max. string voltage in this battery.
16	monStringMaxTemp	INTEGER	Max. string temperature in this battery.

monStringTable (1408)

The string table shows information about each string in the system, using a unique index value for each row (column 1).

The complete OID to the monStringTable is:

1.3.6.1.4.1. 11504.1.1.1408

And so to request the first row of the monStringTable using a GET NEXT command the OID would be:

1.3.6.1.4.1. 11504.1.1.1408.1.1

Identifier	Name	Data Type	Description
1	monStringIndex	INTEGER	A number identifying this string in the system.
2	monStringName	DisplayString	String name
3	monStringStatus	INTEGER	String Status – see start of tables section for details.
4	monStringVoltage	INTEGER	String voltage
5	monStringCurrent	INTEGER	String current
6	monCellAvgVoltage	INTEGER	Average cell voltage
7	monCellAvgOhmic	INTEGER	Average cell resistance
8	monCellMinVoltage	INTEGER	Minimum cell voltage
9	monCellMinOhmic	INTEGER	Minimum cell resistance
10	monCellMaxVoltage	INTEGER	Maximum cell voltage
11	monCellMaxOhmic	INTEGER	Maximum cell resistance
12	monCountJarsInThermalWarning	INTEGER	The count of jars that are in thermal warning on this string.



13	monThermalDisconnectDateTime	DisplayString	<p>The time and date when a thermal runaway disconnect occurred on this string. The interpretation of this register depends upon several factors. If thermal protection is not enabled for this string or no thermal runaway conditions have been detected, this value will be set to null - "".</p> <p>If the string has thermal runaway conditions, monStringStatus (bit #7 ON), but has not yet been disconnected (bit #8 OFF) then these registers show the time in the future when the disconnect relay for this string will be energized. If the string has already been disconnected, monStringStatus (bit #8 ON), then this shows the time in the past when this string was disconnected. Even once the user re-arms this string, the time will still reflect the time it was last disconnected, so the user should always consult monStringStatus bits #7 and #8 to determine the meaning of this time.</p> <p>field octets contents range</p> <p>-----</p> <table> <tr> <td>1</td><td>1-2</td><td>year*</td><td></td></tr> <tr> <td colspan="4">0..65536</td> </tr> <tr> <td>2</td><td>3</td><td>month</td><td>1..12</td></tr> <tr> <td>3</td><td>4</td><td>day</td><td>1..31</td></tr> <tr> <td>4</td><td>5</td><td>hour</td><td>0..23</td></tr> <tr> <td>5</td><td>6</td><td>minutes</td><td>0..59</td></tr> <tr> <td>6</td><td>7</td><td>seconds</td><td>0..60</td></tr> <tr> <td colspan="4">(use 60 for leap-second)</td> </tr> <tr> <td>7</td><td>8</td><td>deci-seconds</td><td>0..9</td></tr> </table> <p>* Notes:</p> <p>-the value of year is in network-byte order</p> <p>For example, Tuesday May 26, 1992 at 1:30:15 PM EDT would be displayed as: 1992-5-26,13:30:15.0</p>	1	1-2	year*		0..65536				2	3	month	1..12	3	4	day	1..31	4	5	hour	0..23	5	6	minutes	0..59	6	7	seconds	0..60	(use 60 for leap-second)				7	8	deci-seconds	0..9
1	1-2	year*																																					
0..65536																																							
2	3	month	1..12																																				
3	4	day	1..31																																				
4	5	hour	0..23																																				
5	6	minutes	0..59																																				
6	7	seconds	0..60																																				
(use 60 for leap-second)																																							
7	8	deci-seconds	0..9																																				
14	monThermalProtectionEnabled	INTEGER enabled(1), disabled(2)	Indicates if thermal runaway protection is enabled(1) or disabled(2)																																				
15	monThermalVoltageLimit	INTEGER	The thermal runaway voltage level set for jar (V) * 10																																				



16	monThermalTempLimit	INTEGER	The thermal runaway temperature limit for this string.(Temperature units) * 10
17	monHighStringVoltageLimit	INTEGER	String voltage high threshold alarm (V) * 10
18	monLowStringVoltageLimit	INTEGER	String voltage low threshold alarm (V) * 10



monJarTable (1792)

The jar table shows information about each jar in the system, using a unique index value for each row (column 1).

The complete OID to the monJarTable is:

1.3.6.1.4.1. 11504.1.1.1792

As described in the previous section, to request the first row of this table using the GET NEXT command the OID would be:

1.3.6.1.4.1. 11504.1.1.1792.1.1

Identifier	Name	Data Type	Description
1	monJarIndex	INTEGER	A unique reference to this jar in the system
2	monJarBatteryNo	INTEGER	A unique reference to the battery of which this jar is a part.
3	monJarBatteryName	DisplayString	The name of the battery of which this jar is part
4	monJarStringNo	INTEGER	A reference to the string of which this jar is a part. This belongs to the battery referenced by monBatteryNo.
5	monJarStringName	DisplayString	The name of the string of which this jar is a part.
6	monJarLabel	DisplayString	The name of this jar or jars as identified by Cellwatch. This may span more than 1 jar if more than one jar is monitored per channel.
7	monJarBlkFr	INTEGER	This starts at 1 and counts upward, typically representing the cell labels on the string
8	monJarBlkTo	INTEGER	Same as monBlkFr if only one jar per channel is measured
9	monJarVoltage	INTEGER	Jar/block voltage (V) * 100 E.g. If monJarVoltage = 1224, then the actual jar voltage is 12.24V
10	monJarOhmic	INTEGER	Jar/block ohmic value (mOhms) * 1000 E.g. if monJarOhmic = 567, then the ohmic value is 0.567 mOhms
11	monJarHVAAlarm	INTEGER	Jar/block voltage high threshold alarm (V) * 100
12	monJarLVAAlarm	INTEGER	Jar/block voltage low threshold alarm (V) * 100
13	monJarHZAAlarm	INTEGER	Jar/block ohmic value high threshold alarm (mOhms) * 1000
14	monJarLZAAlarm	INTEGER	Jar/block ohmic value low threshold alarm (mOhms) * 1000



monAlarmHistoryTable (1803)

The alarm history table shows information about the most recent Cellwatch alarms recorded in the system. The table is a list of the most recent events beginning with the most recent event listed in monAlarmHistoryIndex identified below. This is representative of the 20 most recent alarms parsed into the Data Manager on the IBMU sorted from most recent to least. Each event will have its own unique ID as defined by the event database located on the IBMU. These events match what is visible in the DataManager.

The complete OID to the monAlarmHistoryTable is:

1.3.6.1.4.1. 11504.1.1.1803

As described in the previous section, to request the first row of this table using the GET NEXT command the OID would be:

1.3.6.1.4.1. 11504.1.1.1803.1.1

Identifier	Name	Data Type	Description
1	monAlarmHistoryIndex	INTEGER	A unique reference to the most recent alarm in Cellwatch
2	monAlarmId	INTEGER	A unique identifier that matches the Alarm_ID in DataManager
3	monAlarmSerial	DisplayString	This is most often going to be the same as above, except in cases where users have placed a prefix before the unique ID. e.g. 321 may be viewed as CW321
4	monAlarmBatteryName	DisplayString	The name of the battery in which the alarm occurred
5	monAlarmStringName	DisplayString	The name of the string in which the alarm occurred
6	monAlarmLabel	DisplayString	The jar (voltage or ohmic) or probe (current or temperature) on which the alarm occurred
7	monAlarmType	DisplayString	The alarm type
8	monAlarmValue	INTEGER	The value that triggered the alarm
9	monAlarmUnits	DisplayString	The units associated with an alarm value
10	monAlarmHighLimit	INTEGER	The high threshold limit that was crossed for the high alarm
11	monAlarmLowLimit	INTEGER	The low threshold limit that was crossed for the low alarm
12	monAlarmStart	INTEGER	The starting date and time of the alarm based on a UNIX time stamp, i.e. the number of seconds since 1970/01/01, representative of the time the event occurred on the IBMU



13	monAlarmEnd	INTEGER	<p>Value will be '-1' if the alarm is still active.</p> <p>Values greater than 0 will indicate a UNIX time stamp, i.e. the number of seconds since 1970/01/01, indicating the end time of the alarm.</p>
----	-------------	---------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------



Traps

The following table shows the traps which will be sent out by the SNMP Agent and the events which trigger them.

The specific type is relative to the **alerts** section of the iBMU MIB which has the following OID:

1.3.6.1.4.1. 11504.1.2

(1.3.6.1.4.1.NDSL.BMU.ALERTS)

Specific Type	Trap Name	Event	Variable Bindings
Alarm Events			
These traps are sent in response to Cellwatch detecting that an alarm has started or ended.			
1	monScanningStart	The user starts the Cellwatch software scanning. Log file 09:11:59 System started.	monSysName
2	monScanningStop	The user stops Cellwatch software scanning Log file 15:45:47 System stop requested	monSysName
3	monAlarmVoltageHigh	Sent when a voltage high alarm starts on a jar.	monSysName monAlarmBattery monAlarmString monAlarmJar monValueVoltage
4	monAlarmVoltageLow	Sent when a voltage low alarm starts on a jar.	monSysName monAlarmBattery monAlarmString monAlarmJar monValueVoltage
5		UNUSED	



6	monDischargeStart	<p>Triggered once a discharge starts on a battery (this is usually 5 seconds after the current alarm is triggered – but this delay can be configured for each Cellwatch system in the Cellwatch ini file).</p> <p>Log file: discharge commenced on Battery:</p>	sysName monAlarmBattery
7	monDischargeEnd	<p>Triggered once a discharge has ended on a battery.</p> <p>Log file: Discharge finished on Battery:</p>	sysName monAlarmBattery
8	monAlarmOhmicHigh	<p>Sent when a high ohmic alarm is detected on a string.</p>	monSysName monAlarmBattery monAlarmString monValueOhms
9	monAlarmOhmicLow	<p>Sent when a low ohmic alarm is detected on a string.</p>	monSysName monAlarmBattery monAlarmString monValueOhms
10	monAlarmTempStart	<p>Sent when delayed temperature alarm is detected on a battery.</p> <p>Log file: Delayed temperature alarm on Battery:</p>	monSysName monAlarmBattery



11	monAlarmTempEnd	<p>Sent when delayed temperature alarm has ended on a battery.</p> <p>Log file: Cleared alarm - Delayed Temperature on Battery:</p>	monSysName monAlarmBattery
20	monAlarmBatteryEnd	Sent when all alarms have cleared on a battery.	monSysName monAlarmBattery
21	monAlarmStringVoltageHigh	A high string voltage alarm has occurred on a string.	monAlarmBattery, monAlarmString, monValueVoltage
22	monAlarmStringVoltageLow	A low string voltage alarm has occurred on a string.	monAlarmBattery, monAlarmString, monValueVoltage
23	monAlarmThermalRunwayWarning	The Cellwatch system has identified that a string that has exceeded its thermal runaway threshold. This string will be disconnected within 12 hours, unless delayed by the user. If delayed, the string will be disconnected, at most, within 24 hours of the start of the TRS alarm.	monAlarmBattery, monAlarmString, monValueTemp
24	monAlarmThermalRunwayWarningEnd	A thermal runaway warning ended on a string.	monAlarmBattery, monAlarmString
27	monAlarmThermalRunwayDisconnect	The Cellwatch system has identified and disconnected a string that is in thermal runaway.	monAlarmBattery, monAlarmString, monAlarmJar, monValueVoltage



Alarm Settings			
These traps are sent response to a user changing the Cellwatch configuration.			
30	monSetCellThresholds	User has changed cell alarm settings	monSysName monAlarmBattery monAlarmString monAlarmJar monValueOhmsHigh monValueOhmsLow monValueVoltageHigh monValueVoltageLow
31	monSetStringThresholds	User has changed string alarm settings	monSysName monAlarmBattery monAlarmString monValueAmps monValueOhmsHigh, monValueOhmsLow monValueVoltageHigh monValueVoltageLow monValueTempHigh monValueTempLow monValueStringVoltageHigh monValueStringVoltageLow
32	monSetBatteryThresholds	User has changed battery alarm settings	monSysName monAlarmBattery monValueAmps monValueOhmsHigh, monValueOhmsLow monValueVoltageHigh monValueVoltageLow monValueTempHigh monValueTempLow monValueStringVoltageHigh monValueStringVoltageLow
33	monSetSystemThresholds	User has changed system alarm settings	monSysName monValueAmps monValueOhmsHigh, monValueOhmsLow monValueVoltageHigh monValueVoltageLow monValueTempHigh monValueTempLow monValueStringVoltageHigh monValueStringVoltageLow
34	monSetThermalRunaway	Indicates if thermal runaway settings have been changed for a string.	monAlarmBattery, monAlarmString, monValueThermalProtection monValueVoltage, monValueTemp



Event Reporting

These traps are sent when either a scheduled or user initiated event starts (or ends) on the system, such as an ohmic or voltage scan.

50	monScanOhmicStart	Triggered when an ohmic scan starts on a battery.	monSysName monAlarmBattery
51	monScanOhmicEnd	Triggered when all ohmic scans are complete.	monSysName
52	monScanVoltageStart	Triggered when a voltage scan starts on a battery.	monSysName monAlarmBattery
53	monScanVoltageEnd	Triggered when all voltage scans are complete.	monSysName
Testing			
99	monTestTrap	<p>This is a test trap which can be sent by the SNMP Agent to test that the UDP trap recipients are configured correctly.</p> <p>Click the <i>Send test trap</i> button in the Traps tab of the Cellwatch SNMP Agent.</p>	

Trap Variables

These objects are used solely by traps (as variable bindings) to provide supporting information such as data values and alarm thresholds.

Identifier	Name	Data Type	Description
100	monAlarmBattery	DisplayString	The name of the battery.
102	monAlarmString	DisplayString	Name of the string which caused alarm.
104	monAlarmJar	INTEGER	The jar number within a string which caused alarm.
105	monValueVoltage	DisplayString	Identifies voltage that tripped alarm.
106	monValueTemp	DisplayString	Identifies temperature in °C that tripped the alarm.



107	monValueOhms	DisplayString	Identifies ohmic value that tripped alarm
108	monValueAmps	DisplayString	Identifies a value in amps that tripped alarm
109	monValueVoltageHigh	DisplayString	The high voltage alarm threshold that was set.
110	monValueVoltageLow	DisplayString	The low voltage alarm threshold that was set.
111	monValueTempHigh	DisplayString	The high temperature alarm threshold that was set.
112	monValueTempLow	DisplayString	The low temperature alarm threshold that was set.
113	monValueOhmsHigh	DisplayString	The high ohmic alarm threshold that was set.
114	monValueOhmsLow	DisplayString	The low ohmic alarm threshold that was set.
115	monValueStringVoltageHigh	DisplayString	The high string voltage alarm threshold that was set.
116	monValueStringVoltageLow	DisplayString	The low string voltage alarm threshold that was set.
117	monValueThermalProtection	INTEGER { enabled(1), disabled(2) }	Indicates if thermal runaway protection is enabled(1) or disabled(2).