Overview of SNMP and using third party SNMP tools

TEC/NOT/058

There are a number of open standard network protocols that can be used to discover, manage, and debug networked based Flight Test Instrumentation (FTI) systems. One such protocol is Simple Network Management Protocol (SNMP), which is a standard IP network protocol that can be used to interrogate, query, and configure SNMP-enabled devices such as the NET/REC/XXX or SSR/CHS/XXX recorder, or the KAD/BCU/140 Ethernet controller. You can also carry out other tasks such as turning on recordings and setting Events via SNMP.

To describe how SNMP operates and how it can be used to interact with SNMP-enabled Curtiss-Wright devices, this paper is divided into the following sections:

32.1 Overview of the SNMP

SNMP is a simple but powerful protocol that can be used to discover (GET) and configure (SET) information in SNMP-enabled devices. SNMP is a component of the Internet Protocol Suite as defined by the Internet Engineering Task Force (IETF). It consists of a set of standards for network management, including an application layer protocol, a database schema, and a set of data objects. SNMP exposes management data in the form of variables on the managed systems, which describe the system configuration. Variables on the network end node can then be read (GET) or configured (SET) by a remote station. The end node is controlled by the remote computer. In addition, an end node may be configured to set a trap (TRAP) for key events without the need for polling.

SNMP uses an extensible and customizable Management Information Base (MIB) to describe the variables that may be accessed in the FTI device including the structure, interpretation and read/write attributes of the supported variables. The MIB contains global variables that are common to all networked FTI devices.

In networked FTI, SNMP may be used to monitor and remotely configure network nodes such as:

- Data Acquisition Units (DAUs)
- Switches
- PTP Grandmasters
- Network Recorders

Each of these classes of device has a set of specialized discoverable variables that are also described in the MIB.

An SNMP-managed network consists of three key components as shown in the following figure:

- **Network Management System (NMS):** A software application that monitors and controls managed devices in the networked FTI system. NMSs provide the bulk of the processing and memory resources required for network management. There may be one or more NMSs managing the network. The NMS uses User Datagram Protocol (UDP) port 162 for SNMP.
- **Managed device:** A managed network node (DAU, switch, PTP Grandmaster, network-recorder etc.) that contains an SNMP agent. Managed devices collect and store management information and make this information available to NMSs via the SNMP protocol.
- **Agent:** A piece of software which handles SNMP requests that resides in the managed device. An agent has local knowledge of management information and translates that information into a form compatible with SNMP. The agent uses UDP port 161 for SNMP.
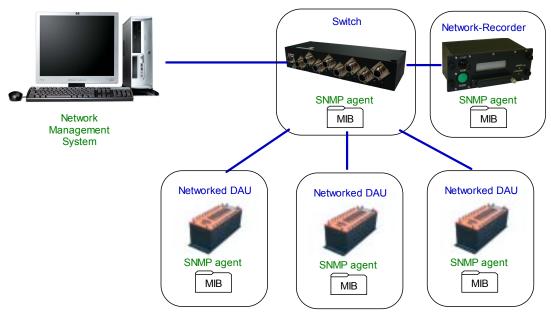
Figure 32-1: An SNMP-managed network

The SNMP protocol is a request-response protocol whereby the NMS issues queries and configures commands via SNMP to the managed device, for example the SNMP messages described in Table 32-1 on page 3 and the SNMP-enabled DAU as shown in the following figure. (See the Microsoft® TechNet SNMP web page for an overview on this topic.)
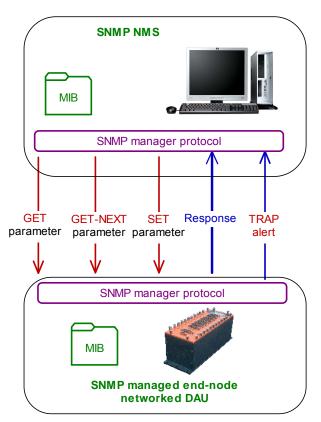

Figure 32-2: SNMP messages in action

Table 32-1: SNMP messages

SNMP version	SNMP command	Description of messages used by the NMS
V1	GET	Retrieves the value of one or more object instances from an agent. For example, GET EventNumber.
V1	GETNEXT	Retrieves the value of the next object instance in a table or a list within an agent. For example, GETNEXT NextEventNumber.
V1	SET	Sets the values of object instances within an agent. For example, SET IPAddress.
V1	TRAP	Asynchronously informs the NMS of a significant event. For example, TRAP TemperatureExceedsThreshold.
V2c	GETBULK	Efficiently retrieves large blocks of data.

32.1.1 Comments on SNMP usage

It is clear that SNMP is a powerful protocol for reconfiguring and querying devices for key settings. However, there are some caveats that must be considered before it is implemented.

- Although TCP transport is possible, SNMP typically runs over UDP, which is a connectionless protocol that does not provide packet delivery acknowledgments or retransmit lost packets. Therefore, there can be no guarantee that commands/messages issued by SNMP are correctly propagated through the network. To mitigate this, each SNMP-SET operation should be safeguarded by a subsequent SNMP-GET operation to verify the success of the SET.
- SNMP allows the NMS to potentially configure critical variables that alter the devices' configuration and operation. Care should be taken when determining the variables that can be set via SNMP. These variables should not interrupt packet switching, data acquisition or allow the device to enter an unstable state. Such critical variables are protected from accidental SNMP SETs using the locking mechanism described in "32.5.2 Setting protected variables using SNMP with Curtiss-Wright devices" on page 15.
- There are several flavors of the SNMP protocol. As currently specified, SNMPv2 is incompatible with SNMPv1 in two key areas: message formats and protocol operations. SNMPv2c messages use different header and Protocol Data Unit (PDU) formats from SNMPv1 messages. SNMPv2c also uses two protocol operations that are not specified in SNMPv1. However, RFC1908 defines two possible SNMPv1/v2c coexistence strategies: proxy agents and bilingual network-management systems.
- SNMPv1 and SNMPv2c pose a security risk in that packet sniffing can be used to monitor auto-discovery advertisements. Although this has been addressed in SNMPv3 through the introduction of encryption techniques, SNMPv3 has a significantly larger footprint and consumes onerous processing resources.

32.2 MIBs

SNMP is a protocol and does not define which information or variables are managed. The variables accessible via SNMP are organized in hierarchies with meta-data (type and variable description). The variables accessible via SNMP are described by MIB files. MIBs describe the structure of the management data of a device subsystem using a hierarchical namespace containing Object IDentifiers (OID) as shown the following figure. Each OID is unique and identifies a variable that can be read or set via SNMP. The following types of managed objects exist:

- Scalar objects define a single object instance.
- Tabular objects define multiple related object instances that are grouped in MIB tables.

One can either access the hierarchy solely via dot-delimited numbers, dot-delimited strings, or a combination of both. For example, the three object identifiers below are all equivalent:

```
.1.3.6.1.4.1.33698.10.21.0

.iso.org.dod.internet.private.enterprises.acra.recorder.controlMethod.0

.iso.org.dod.internet.private.enterprises.33698.10.21.0
```

NOTE: A full listing of the values that can appear in the OID node that comes after *enterprises* can be found at: http://www.iana.org/assignments/enterprise-numbers

A useful feature of the NMS is its ability to perform a MIB or SNMP walk whereby the walk can retrieve a sub-tree of the available variable hierarchy from a specified OID branch and the values returned to the NMS. The MIB walk is achieved using SNMP-GETNEXT messages. For example, by instructing an SNMP walk to start at .iso.org.dod.internet.private.enterprises.acra the variables in the acra subtree can be listed, ignoring any variables which might exist under any of the nodes which are not highlighted in the following diagram:
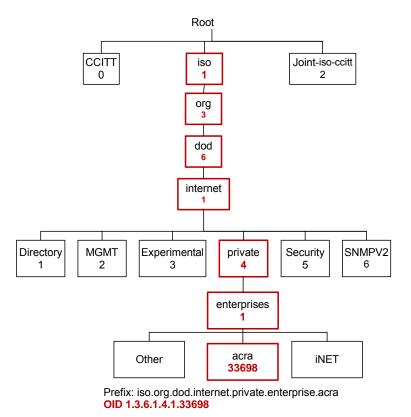

Prefix: iso.org.dod.internet.private.enterprise.acra
OID 1.3.6.1.4.1.33698

Figure 32-3: OID hierarchy

32.2.1 Reading the MIB file

The MIB file describes the managed objects or variables and their properties that can be accessed via SNMP. The Abstract Syntax Notation One (ASN.1) is a standard formal and flexible notation used to describe data structures for representing, encoding, transmitting and decoding data. ASN.1 provides a set of formal rules for describing the structure of objects that are independent of machine-specific encoding techniques and is a precise, formal notation that removes ambiguities.

The MIB file is written using the Structure of Management Information (SMI) notation, which is an adapted subset of the ASN.1.

SMI subdivides into the following three parts:

• **Module definitions**: Describes information modules. An ASN.1 macro, MODULE-IDENTITY, is used to concisely convey the semantics of an information module.
• **Object definitions:** Describes managed objects. An ASN.1 macro, OBJECT-TYPE, is used to concisely convey the syntax and semantics of a managed object.
• **Notification definitions:** (aka *traps*) Describes unsolicited transmissions of management information. An ASN.1 macro, NOTIFICATION-TYPE, concisely conveys the syntax and semantics of a notification.

32.2.1.1 Example of reading the MIB file

The following is an extract from the Acra MIB. All variables (or managed objects) and their hierarchy are described in the MIB. The managed object hierarchy and its mapping to OID is described in the MIB. For example, the following line indicates that the Acra sub-tree (belonging to the Curtiss-Wright Acra Business Unit, formerly ACRA CONTROL Ltd.) is associated with an OID value of 33698 under the enterprises branch:

```
acra OBJECT IDENTIFIER ::= { enterprises 33698 }
```

This is the trunk of the main Acra tree. It starts out with defining the main subtree **acra** as under the enterprises node and identified as sub-node 33698. So, the full path so far is `iso.org.dod.internet.private.enterprises.33698`.

Navigating into the Acra sub-tree, the following hierarchy is in place:

```
devices OBJECT IDENTIFIER ::= { acra 1 }

settings OBJECT IDENTIFIER ::= { acra 2 }

...

recorder OBJECT IDENTIFIER ::= { acra 10 }

...
```

Therefore, to reach the Acra recorder sub-tree, the OID is:

```
iso.org.dod.internet.private.enterprises.acra.recorder

1  .3  .6  .1      .4      .1          .33698.10
```

The MIB also defines the managed objects that may be accessed via SNMP. By way of illustration, using an SNMP variable definition within the MIB, consider the variable defined as the `controlMethod`. This `controlMethod` variable is used with the NET/REC/XXX and SSR/CHS/XXX products to configure how a recorder is to be controlled. Three control methods are possible locally using the front panel START/STOP and EVENT buttons (`local, 0`); using SNMP commands over the network (`network, 1`); or remotely using discrete I/O signals (`remote, 2`). The `controlMethod` object has an `integer` value with three possible values relating to the three control methods. This variable has `read-write` access so SNMP-SET and SNMP-GET commands can be used to retrieve and modify the value of this variable. The default value, `DEFVAL`, is `network`, as seen in the following code:

```
controlMethod  OBJECT-TYPE

SYNTAX     INTEGER {

local (0),

network (1),

remote (2)

}

MAX-ACCESS read-write

STATUS     current

DESCRIPTION

"Method of controlling the recorder.
Only the control method listed in this variable shall control the recorder
with  the  exception  that  the  controlMethod  variable  itself  shall  always  be
modifiable with SNMP.
(0) local - controlled by front panel,
(1) network - controlled over the network via SNMP,
(2) remote – discrete I/O signal controlled from cockpit."

DEFVAL  { network }

::= { recorder 21 }
```

The final entry in this MIB entry is the location of the variable within the MIB hierarchy. In this case the `controlMethod` variable is located at subtree index 21 within the recorder branch of the MIB:

iso	org	dod	internet	private	enterprises	acra	Recorder	ControlMethod
1	3	6	1	4	1	33698	10	21

The final part of the OID indicates whether the variable is a scalar object, a tabular object, or an array object. Scalar variables describe a single object instance, whereas tabular variables describe multiple related instances. For example, `controlMethod` is scalar, because there is only one variable in the recorder which determines how the recorder is controlled. The description of an Ethernet port (`iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifDescr`) is tabular because a device may have multiple Ethernet ports.

In the OID for a scalar variable, the final part of the OID is 0. In a tabular OID, the final part is the index, with 1 indicating the first item in the table. So, for example, a switch with eight ports would have variables `...ifEntry.ifDescr.1` to `...ifEntry.ifDescr.8` giving descriptions of the eight ports.

Since `controlMethod` is scalar, the index value is 0.

The syntax of the path for this OID is:

```
.iso.org.dod.internet.private.enterprises.acra.recorder.controlMethod.0
.1.3.6.1.4.1.33698.10.21.0
```

The MIB sections list out in great detail the hierarchy under the acra subtree; time, devices, settings, identification, and so on. It's not necessarily in the order above. Nor are all the sections actually supported by all actual devices. Since all the sections are similar, only some example subtree sections are discussed below.

Settings subtree section

The Settings section is where most of the important variables are defined. One of the most common uses of SNMP is to set the IP address.

```
ipSetVar OBJECT-TYPE
        SYNTAX   IPSetVarType
        ACCESS   read-write
        STATUS   current
        DESCRIPTION
"Combined MAC address, IP address and IP mask"
        ::= { settings 1 }
```

In this case, we are looking at the ipSetVar node, which shows the MAC, IP address and IP mask together in one variable. It can be accessed by the full path of `.iso.org.dod.internet.private.enterprises.33698.2.1.0`.

In a recorder, you might want to increment the Event number, by setting the `triggerRemoteEvent` variable. To do this, you have to go through the triggers node:

```
.iso.org.dod.internet.private.enterprises.33698.3.10.1.0
        -- Triggers
        recorderTriggers OBJECT IDENTIFIER ::= { triggers 10 }
        triggerRemoteEvent OBJECT-TYPE
        SYNTAX INTEGER (0..1)
        ACCESS read-write
        STATUS current
        DESCRIPTION
"Event increment variable, writing 1 to this variable will increment the event number reading from this variable returns the current event number"
        DEFVAL { 0 }
        ::= { recorderTriggers 1 }
```

Time subtree section

```
Subtree under acra.time
ptp2            OBJECT IDENTIFIER ::= { time 9 }
```

Since time is subtree acra.13, then ptp2 is subtree acra.13.9.

Ptp2 doesn't show up on all devices, even networks switches; the NET/SWI/004/EM1, for example, does not support PTPv2, and therefore does not have any variables in the Ptp2 subtree.

Device Identifiers section

This section lists every device Curtiss-Wright supports with SNMP, as of the time the version of the MIB file was released. To see the description, look at the entries under acra.devices.

NOTE: Devices from Acra use the acra.devices OIDs to specify an identifier for each device type. A more descriptive variable, which lists version numbers, is in:
.iso.org.dod.internet.mgmt.mib-2.system.sysDescr.0.

32.3 SNMP and the MIB in action

To demonstrate SNMP in action, the example of SNMP as shown in the following figure is described. Consider a system comprised of an NMS and an SNMP-enabled DAU, where the DAU has a hypothetical SNMP-Trap variable for temperature. The NMS sends an SNMP-GET message to the DAU to retrieve the value for the temperature threshold variable as defined in the MIB supported by the DAU. The DAU's own SNMP-Agent interprets the received SNMP-GET message containing the OID of the temperature threshold variable. In this way, the DAU SNMP-Agent can retrieve the desired variable value and return the appropriate response to the requesting NMS. Should the NMS choose to reconfigure the temperature threshold variable on the DAU, the NMS sends an SNMP-SET message to the DAU. However the temperature threshold variable must have read/write attributes so that it can be reconfigured. The SNMP-Agent continuously monitors its current temperature over time. When the DAU detects that the current temperature exceeds the newly defined temperature threshold, the DAU can send an SNMP-Trap to the trap listener in the NMS to warn it of the occurrence of this event.
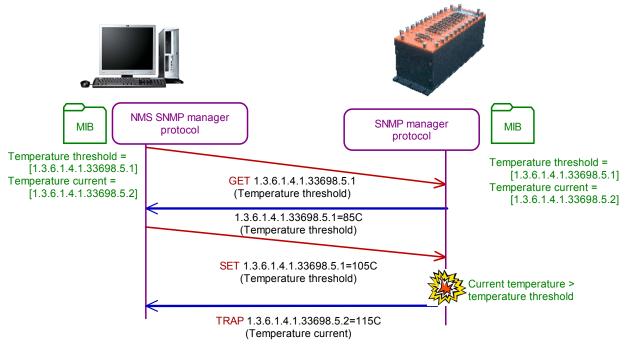

Figure 32-4: SNMP example

32.4 SNMP software

There is much free and commercial SNMP software available to perform SNMP tasks including:

- Network Monitoring Software - Loriot Pro, free and commercial versions available.
- OpManager – ManageEngine, commercial version.
- NetDecision – NetMechanica, commercial version.
- Network Management Tools – SimpleSoft, commercial version.
- MIB Browser and SNMP Manager - iReasoning, commercial.
- Net-SNMP- free C/C++ open source API and binary.
- Netsnmpj - java open source implementation available on SourceForge.
- JMIBBrowser - java open source implementation available on SourceForge.
- Frameflow MIB Browser -Frameflow is a good way of exploring the SNMP hierarchy to get a feel for the structure. It allows read-only access, so you cannot accidentally change important values.
- kSNMP - This is a Curtiss-Wright in-house generated software program created to add a GUI interface to the SNMP settings. It has been superseded by DAS Studio 3, but is still of value to KSM-500 users. Contact Curtiss-Wright support (acra-support@curtisswright.com) for more information.

- DAS Studio 3 - This is the main Curtiss-Wright software used to control and query all Curtiss-Wright hardware, including DAUs, switches, and recorders. It provides a GUI interface to the entire system. For SNMP devices, it allows access to the normal control variables as well as more advanced features like filtering or X-Bar setup. Contact Curtiss-Wright support (acra-support@curtisswright.com) for information.

32.4.1 Installation and setup of Net-SNMP

This section describes using the Net-SNMP open source utility to perform SNMP tasks with Curtiss-Wright devices.

1. Download and install the Net-SNMP library according to the library's instructions.

NOTE: The latest Net-SNMP library can be downloaded here: http://www.net-snmp.org/download.html, which points to http://sourceforge.net/projects/net-snmp/files/.

2. Ensure that the appropriate MIBs are installed in the correct location for Net-SNMP to access them (contact Curtiss-Wright support [acra_support@curtisswright.com] for the latest files). To add the Acra MIB, the iNET MIB and any other relevant MIBs to Net-SNMP, first copy the MIBs to the following directory:

```
<installation root>:\NetSNMP\share\snmp\mibs
```

3. Then register the newly added MIBs by modifying the following config file:

```
<installation root>:\NetSNMP\etc\snmp\snmp.conf
```

4. Add the following lines to the config file:

```
mibs +BRIDGE-MIB

mibs +EtherLike-MIB

mibs +ACRA-MIB

mibs +TMNS-MIB
```

ACRA-MIB is Curtiss-Wright's Acra MIB and TMNS-MIB is the iNET MIB.

5. The config file should now contain the following:

```
mibdirs C:/NetSNMP/share/snmp/mibs

persistentDir C:/NetSNMP/snmp/persist

tempFilePattern C:/NetSNMP/temp/snmpdXXXXXX

mibs +BRIDGE-MIB

mibs +EtherLike-MIB

mibs +ACRA-MIB

mibs +TMNS-MIB
```

32.4.2 Using Net-SNMP

The Net-SNMP utility can be run from the command line to:

- Retrieve information from an SNMP-capable device, either using single requests (`snmpget, snmpgetnext`), or multiple requests (`snmpwalk, snmptable, snmpdelta`).
- Manipulate configuration information on an SNMP-capable device (`snmpset`).
- Retrieve a fixed collection of information from an SNMP-capable device (`snmpdf, snmpnetstat, snmpstatus`).
- Convert between numerical and textual forms of MIB OIDs, and display MIB content and structure (`snmptranslate`).

In this section, only the basic SNMP-SET and SNMP-GET commands are described.

The basic syntax for SNMP-SET commands is as below:

```
snmpset [Version] [Community][Agent] [OID] [Type] [Value]
```

Field	Typical values	Description
Version	-v 2c	Specifies SNMP version to use. All Curtiss-Wright devices support SNMPv2c.

Field	Typical values	Description
Community	-c public	SNMP Version 1 or 2c specific command line field. The "community string" allows the authentication of clients, in effect a type of password, which is transmitted in clear text.
Agent	192.168.28.1	IP address of the managed device. Be sure the NMS is on the same subnet.
OID	OID in text	OID of the target variable, for example: `.iso.org.dod.internet.private.enterprises.acra.recorder.controlMethod`
Type	i	One of i, u, t, a, o, s, x, d, b, U, I, F, or D i: INTEGER u: unsigned INTEGER t: TIMETICKS a: IPADDRESS o: OBJID s: STRING x: HEX STRING d: DECIMAL STRING b: BITS U: unsigned int64 I: signed int64 F: float D: double
Value	1	The appropriate value associated with the preceding type.

For example:

```
snmpset -v 2c -c public 192.168.28.1 .iso.org.dod.internet.private.enterprises.acra.recorder.controlMethod.0 i 2
```

(as a single line, with no line break) is the command line used to issue an SNMP-SET command to a NET/REC/001 with an IP address of 192.168.28.1 to change the setting of the `controlMethod` variable to an integer value of 2, where a value of 2 is mapped to the remote control. Similarly, the SNMP-GET command has a similar format:

```
snmpget -v 2c -c public 192.168.28.1 .iso.org.dod.internet.private.enterprises.acra.recorder.controlMethod.0
```

Following are some more examples.

Get the system name

```
snmpget -v 2c -c public 192.168.28.4 .iso.org.dod.internet.mgmt.mib-2.system.sysName.0
```

might return `SNMPv2-MIB::sysName.0 = STRING: Left Wing DAU`.

Change the system name

```
snmpset -v 2c -c public 192.168.28.4 .iso.org.dod.internet.mgmt.mib-2.system.sysName.0 s The_System_Name
```

which will return `SNMPv2-MIB::sysName.0 = STRING: The_System_Name`.

Change the IP address on a KAD/BCU/140/x module

1. Verify the current IP address with ping or Wireshark® (below), and use it for the steps below.
2. Run these SNMP commands: (Note, copy/paste into a command shell may not work correctly; for example the '-' character may be replaced by a Unicode dash symbol.) This example shows changing from 192.168.28.1 to 192.168.28.5:
   a. Set the BCU module to Change IP Address mode.
      ```
      snmpset -v 2c -c public 192.168.28.1 iso.org.dod.internet.private.enterprises.acra.settings.enableFailSafeIPSet.0 i 1
      ```
   b. Show a hex string with the current MAC address, current IP address and current IP Mask.
      ```
      snmpget -v 2c -c public 192.168.28.1 .iso.org.dod.internet.private.enterprises.acra.settings.ipSetVar.0
      ```
      Example return value: HEX-STRING: 00 00 0C 4D 8D 61 FA **C0 A8 1C 01** 00 00 00 00 FF FF FF FF
   c. Change the IP Address. The example below sets it to 192.168.28.05 (C0A81C05 HEX).
      ```
      snmpset -v 2c -c public 192.168.28.1
      ```

```
     .iso.org.dod.internet.private.enterprises.acra.settings.ipSetVar.0 x
     00000C4D8D61FAC0A81C0500000000FFFFFFFF
```

    d. Verify the IP address has been changed.

```
snmpget -v  2c -c public 192.168.28.5 .iso.org.dod.internet.private.enterprises.acra.
settings.ipSetVar.0
```

    e. Exit Change IP Address mode.

```
snmpset -v 2c -c public 192.168.28.5 .iso.org.dod.internet.private.enterprises.acra.
settings.enableFailSafeIPSet.0 i 0
```

3.   Ping the new IP address to test it.

Change the IP address on a NET/SWI/003

1.   Verify the current IP address with ping or Wireshark, and use it for the steps below.
2.   Run these SNMP commands: (Note, copy/paste into a command shell may not work correctly; for example the '-' character may be replaced by a Unicode dash symbol.)

    a.
```
snmpset -v 2c -c public <OLD_IP_ADDRESS>
.iso.org.dod.internet.private.enterprises.acra.settings.enableFailSafeIPSet.0 i 1
```

    b.
```
snmpset -v 2c -c public <OLD_IP_ADDRESS>
.iso.org.dod.internet.private.enterprises.acra.settings.ipSetVar.0 x
00000C4DAC7A00<NEW_IP_ADDRESS_IN_HEX>00000000FFFFF
```
       For example: `00000C4DAC7A00C0A81C0500000000FFFFFFFF`

       Don't change any other characters. They are the MAC address and masks.

    c.
```
c.snmpset -v 2c -c public <NEW_IP_ADDRESS>
.iso.org.dod.internet.private.enterprises.acra.settings.enableFailSafeIPSet.0 i 0
```

Finding SNMP variables

The command `SNMPWALK` recursively goes through the entire tree or subtree, depending on the starting location. For example, `snmpwalk -Of -v 2c -c public 192.168.28.4` displays the full MIB address and value of each valid SNMP variable. Whereas `snmpwalk -Of -v 2c -c public 192.168.28.4 .iso.org.dod.internet.mgmt.mib-2` displays only the MIB-2 tree.

32.5 Appendix - Curtiss-Wright SNMP support

This section lists a subset of the variables supported by Curtiss-Wright devices including the NET/REC/XXX and SSR/CHS/XXX recorders and the NET/SWI/XXX family of products.

The attributes for each of the variables is indicated:

- SET: the value of the variable may be modified via SNMP-SET command.
- GET: the value of the variable may be retrieved via SNMP-GET command.
- NA: the variable is not accessible as it is used to implement a structure on the MIB hierarchy.

Table 32-2: Public sub-tree: .iso.org.dod.internet.mgmt.mib-2

Variable name	Attributes	Description
`.system`	NA	
`.system.sysDescr`	GET	Returns a string containing the name and version identification of the device. For example, on the NET/SWI/004, this returns: "NET/SWI/004 Mar 12 2010 TIC/V/038 A00 TIC/V/037 1600 FWA/W/001 1.02".
`.system.sysObjectID`	GET	Returns one of the values under Devices in the Acra MIB, listed below, identifying the device.
`.system.sysName`	GET/SET	This is a string, which you may set. By default, this is left blank. For example, sysName may be "MyCockpitNetworkRecorder".
`.system.sysLocation`	GET/SET	This is a string, which you may set. By default, this is left blank. For example, sysLocation may be "Cockpit".

Table 32-2: Public sub-tree: .iso.org.dod.internet.mgmt.mib-2

`.system.sysServices`	GET	Return an integer whose bits indicate what parts of the network stack the device mainly works with. Bit 0 represents the physical layer (layer 1), up to bit 6 for the application layer (layer 7). The NET/SWI/004 returns 72, implying Application level (PTP) and Transport level (switching).
Variable name	**Attributes**	**Description**
`.interfaces`	NA	
`.interfaces.ifNumber`	GET	Number of supported Ethernet interfaces. The NET/SWI/004 is an 8-port Ethernet switch and as such reports back a value of 8. This normally coincides with the number of entries in `.interfaces.ifTable`, but an SNMP walk does not depend on this.
`.interfaces.ifTable`	NA	Table of `ifEntry` structures, giving information about the interfaces. These contain the following information: See table below.

Table variable name	Attributes	Description
`.ifEntry.ifIndex.`*n*	NA	Where *n* is the index. Returns the index value, *n.*
`.ifEntry.ifDescr.`*n*	GET	Description of the interface, for example onboard Ethernet.
`.ifEntry.ifType.`*n*	GET	Returns a value indicating the type of interface. The value returned is the one specified for all Ethernet-like interfaces regardless of speed.
`.ifEntry.ifSpeed.`*n*	GET	Returns the speed of the interface in bits per second.
`.ifEntry.ifPhysAddress.`*n*	GET	MAC address of the interface.
`.ifEntry.ifOperStatus.`*n*	GET	Indicates whether interface *n* is operating. Values that may be returned are: up; down; testing; unknown; dormant; notPresent; or lowerLayerDown.

Table 32-2: Public sub-tree: .iso.org.dod.internet.mgmt.mib-2

`.ip`	NA	
`.ip.ipAddrTable`	NA	The table of addressing information relevant to this entity's IP addresses. An interface may have multiple IP addresses or none, so the number of entries in this table is not necessarily the same as that in `.interfaces.ifTable`.

Table variable name	Attributes	Description
`.ipAddrEntry` `.ipAdEntAddr.`*n*	GET	The IP address to which this entry's addressing information pertains.
`.ipAddrEntry` `.ipAdEntIfIndex.`*n*	GET	The index value, which uniquely identifies the interface to which this entry is applicable. The interface, identified by a particular value of this index, is the same as the value of RFC 1573's `ifIndex`. That means its value ranges between 1 and the value of `ifNumber`. The value for each interface must remain constant at least from one re-initialization of the entity's network management system to the next re-initialization.
`.ipAddrEntry` `.ipAdEntNetMask.`*n*	GET	Returns the Netmask, indicating which bits of the IP address refer to the local subnet.

`.host`	NA	
`.host.hrSystem.hrSystemDate`	GET	Return the module's notion of the current local date and time of day, as a `DateAndTime` structure defined in RFC2579.

32.5.1 Acra sub-tree: .iso.org.dod.internet.private.enterprises.acra

Curtiss-Wright's MIB identification string is ACRA-MIB.

NOTE: This section describes only a subset of the SNMP managed variables that are available across the Curtiss-Wright product families. For more details on SNMP-managed variable support for a specific device, refer to the corresponding data sheet and supporting documentation for that device. If you still require information on specific variables, contact Curtiss-Wright support (acra-support@curtisswright.com).

The following data types are defined in the Acra MIB:

- `IPSetVarType`**:** A combination of Interface index, MAC address, IP address, IP mask, and Gateway IP address. This is used when setting these variables on a device. The packet to set this information may be broadcast. Each of these fields must be set since normal point-to-point communication may not be possible if some or all of these fields are not correctly set.
- `acraDeviceType`**:** Used in conjunction with the variable `.identification.networkNodeType` and is described in the following table. The enumerated possible values for the device type are as follows:
  - Null
  - Network device
  - DAU
  - Recorder
  - Time master

Variable name	Attributes	Description
.devices	NA	Defines values that indicate the type of device returned by system.sysObjectID above. Object IDs are generated based on the type number of the device. For a specific device the OID is `.iso.org.dod.internet.private.enterprises.acra.devices.<type number>.0` At present the MIB defines values for NET/REC/001, SSR/CHS/001, NET/REC/002 and NET/SWI/004.
.settings	NA	Returns one of the values under Devices in the Acra MIB, listed below, identifying the device.
.settings.ipSetVar	GET/SET	Combined MAC address, IP address, netmask and Gateway IP address. These are sent in a single SNMP packet as explained above under `.IPSetVarType`.
.settings.serial	GET/SET	Serial number of the device. This is a string with a maximum size of 16 characters.
.settings.factoryConfig	GET	Read-only, indicates whether the module is in Factory Mode.
.settings.enableFailSafeIPSet	GET/SET	Enable or disable write access to `ipSetVar` or related variables. The IP address can only be changed if this is set to 1. The host should reset this value to 0 when the IP address has been changed. However, if the host fails to do so, it reverts to 0 when the device is power cycled or if the variable is not updated and if no operation which depends on it, such as setting the IP address, occurs for five minutes.
.triggers	NA	This sub-tree is used for variables that trigger events.
.triggers.recorderTriggers	NA	Sub-tree containing variables which trigger events on the NET/REC/XXX.
.triggers.recorderTriggers .triggerRemoteEvent	GET/SET	Writing 1 to this variable increments the event number and causes an event packet to be recorded.
.recorder	NA	
.recorder.wrapAround	GET/SET	Controls whether the recording stops when the CompactFlash card is full, or whether the oldest data is overwritten.
.recorder.nextEventNumber	GET/SET	Reads or sets the number of the next Event record.
.recorder.streamSpeed	GET	Current filtered Ethernet packet speed (bytes).
.recorder.currentFileName	GET	Indicates the name of the file currently being written to by the recorder.
.recorder.recMode	GET	Indicates the current mode of the recorder. This may have the following values: • `idle:` Result of stop or mount. • `record:` Record command was received. • `dismounted:` Media is dismounted (this value is not used in the NET/REC/001/02, as mounting is automatic). • `notReady:` Mode used when recorder is being configured or other times when functional control of the recorder is not available. • `mediaNotPresent:` Media not present. • `mediaFull:` Media full. • `mediaLocked:` Media is write-protected.

Variable name	Attributes	Description
.recorder .recorderMulticastGroupsTable	NA	Table of Multicast IP groups that the recorder should join. This allows the recording of multicast data, which would otherwise be blocked by a switch. Elements contain the following: <table><tr><th>Table variable name</th><th>Attributes</th><th>Description</th></tr><tr><td>.groupIndex.n</td><td>GET</td><td>Return the index value, n.</td></tr><tr><td>.IPAddress.n</td><td>GET/SET</td><td>Multicast IP address.</td></tr></table>
.recorder.recCommand	GET/SET	Issues a command to the recorder, or returns the last command issued. Command values are: • noCommand: Returned when recCommand is read, if no command has yet been received. • record: Starts the recording function. • stop: Stops recording function. • mount: Mount the media for read/write access (this value is not used in the NET/REC/001, as mounting is automatic). • dismount: Dismount the media in preparation for removal. Recorder must be in the idle state for this command to be valid. (This value is not used in the NET/REC/001; in the stopped state you can remove the media without dismounting.)
.recorder.controlMethod	GET/SET	Select or read whether the recorder is controlled by its front panel, by SNMP, or by another interface. The controlMethod variable remains available via SNMP even if the local or external method is selected, but other control functions are no longer available. Values of this variable are: • local: Controlled by the front panel. • network: Controlled over the network via SNMP. • external: Controlled from the cockpit over some other interface.
.recorder.erase	GET/SET	Erases the recording media. Set to true to start erase. Recorder sets this value to false when the erase process is complete.
.recorder.defaultFileSize	GET/SET	Default PCAP file size in megabytes or zero to set it automatically based on the media size.
.recorder.freeSpaceWarningLimit	GET/SET	Free space warning limit in megabytes or zero to disable blinking warning.
.display	NA	Controls the front panel display of the recorder.
.display.brightness	GET/SET	Returns or sets the display brightness, as a percentage of the full brightness level.
.time	NA	
.time.curLeapSeconds	GET/SET	Reads or sets the number of leap seconds since the start of the PTP epoch.
.time.curYear	GET/SET	Reads or sets the current year. If the device is synchronized to IRIG, or another time source that does not provide year information, this defines the current year. If the device synchronizes to GPS or PTP, this variable is updated with information from the GPS or PTP source.

Table 32-3: Public sub-tree - .iso.org.dod.internet.private.enterprises.acra

Variable name	Attributes	Description
.time.ptpSyncReliabilityLevel	GET/SET	Reads or sets the definition of "in synch" when synchronizing to a time master. If the PTP stack calculates the current error as having a magnitude, in nanoseconds, less than or equal to this, then the device is considered in synch. In a recorder, this controls the "time reliable" and "time unreliable" event packets recorded on foot of information gathered by the PTP stack.
.time.ptpSyncError	GET	Reads the most recently calculated error in the local clock, that is, the last calculated error relative to the PTP Grandmaster. The value is an integer number of nanoseconds. In the NET/SWI/004, which synchronizes to IRIG or GPS rather than to PTP, this gives the synchronization error relative to the time master in use. If the device is not currently in synch with a time master, the value 999999999 (a nanosecond short of 1 second, and the maximum legal value of a PTP nanoseconds field) is returned.
.time.timeReliable	GET	Reads whether the current time is considered reliable. This returns 1 if the device is receiving time information and the .ptpSyncError value ≤ the .ptpSyncReliabilityLevel value, 0 otherwise.
.time.timeMode	GET/SET	Indicates the current source of time information. Values that can be returned are: • LocalFreeRunning: Not willing to be synchronized to anything. • PTPSlave: Synchronizes to a PTP grandmaster. • LocalTimeNoPTP: Synchronizes to a non-network time source (for example, GPS or IRIG) but does not do PTP. • LocalTimePTPGrandmaster: Synchronize to a non-network time source, and serves out the time as a PTP Grandmaster. • Automatic: This is not yet supported. The device should use the Best Master Algorithm to compare any PTP Grandmaster with the available non-network time source. It becomes master if it has a better clock, otherwise it is a PTP slave.

32.5.2 Setting protected variables using SNMP with Curtiss-Wright devices

The use of SNMP to access and modify configuration settings on devices should be used with care and be protected against accidental or inadvertent modification. SNMP configurable variables that affect the operation of a device are considered to be critical variables. These critical variables are protected by a simple 2-step locking mechanism. To illustrate this locking mechanism, consider the variable ipSetVar that is used to modify the IP address of a given device using SNMP.

32.5.2.1 STEP 1: Unlock access to the critical variable

In the Acra MIB, there is an enableFailSafeIPSet variable located within the Settings branch.

iso	org	dod	internet	private	enterprises	acra	Settings	EnableFailSafeIPSet
1	3	6	1	4	1	33698	2	4

This variable is described in the MIB as follows:

```
enableFailSafeIPSet OBJECT-TYPE
SYNTAX INTEGER { disable(0), enable(1) }
ACCESS read-write
STATUS current
```

```
DESCRIPTION

"Enable/disable IP settings"

DEFVAL { disable }

::= { settings 4 }
```

To unlock access to any critical variable, the variable `enableFailSafeIPSet` must be first enabled. The command used by Net-SNMP to SET this variable to enable is:

```
snmpset -v2c -c public 192.168.28.1. iso.org.dod.internet.private.enterprises.acra
.settings.enableFailSafeIPSet.0 i 1
```

The following figure illustrates the dissected Wireshark output of the SNMP SET command.

```
⊟ Simple Network Management Protocol
     version: v2c (1)
     community: public
  ⊟ data: set-request (3)
     ⊟ set-request
          request-id: 5
          error-status: noError (0)
          error-index: 0
        ⊟ variable-bindings: 1 item
          ⊟ 1.3.6.1.4.1.33698.2.4.0:
               Object Name: 1.3.6.1.4.1.33698.2.4.0 (iso.3.6.1.4.1.33698.2.4.0)
               Value (Integer32): 1
```

Figure 32-5: Dissected SNMP SET Command to Unlock Critical Variable Access

32.5.2.2 STEP 2: Modify the critical variable

Once the critical variable has been unlocked, the IP address can be modified. The IP Address variable is located in the Settings subtree of the Acra MIB.

iso	org	dod	internet	private	enterprises	acra	Settings	ipSetVar
1	3	6	1	4	1	33698	2	1

This variable is described in the MIB as follows:

```
ipSetVar OBJECT-TYPE

        SYNTAX   IPSetVarType

        ACCESS   read-write

        STATUS   current

        DESCRIPTION

"Combined MAC address, IP address and IP mask"

        ::= { settings 1 }
```

The command used by Net-SNMP to SET the IPAddress variable is:

```
snmpset -v 2c -c public 192.168.28.1 .iso.org.dod.internet.private.enterprises.acr
a.settings.ipSetVar.0 s 01000c4d000000c0a80402ffffff00cc09d7b02
```

where the string is a combined MAC address, IP address, and IP mask value.

NOTE: This setting value takes effect if and only if the `enableFailSafeIPSet` variable has been set to enabled, unlocking access to this protected variable.

The following figure illustrates the dissected Wireshark output of the SNMP SET command.

```
⊟ Simple Network Management Protocol
     version: v2c (1)
     community: public
  ⊟ data: set-request (3)
     ⊟ set-request
         request-id: 6
         error-status: noError (0)
         error-index: 0
       ⊟ variable-bindings: 1 item
         ⊟ 1.3.6.1.4.1.33698.2.1.0: 01000c4d000000c0a80402ffffff00c09d7b02
             Object Name: 1.3.6.1.4.1.33698.2.1.0 (iso.3.6.1.4.1.33698.2.1.0)
             Value (OctetString): 01000c4d000000c0a80402ffffff00c09d7b02
```

Figure 32-6: Dissection SNMP SET command to set the IP address variable

To further safeguard these protected critical variables, the `enableFailSafeIPSet` times out after five minutes of inactivity and returns to a disabled state.

32.6 Troubleshooting

1. Ensure the NMS (desktop PC) is on the same IP subnet.
2. Ensure proper command formatting.
3. Use ping to ensure the device is connected.
4. Use `snmpwalk -Of -v 2c -c public <IP ADDRESS>` to test basic SNMP functionality. If it returns with full character descriptions of the various OIDs and their values then Net-SNMP should be working correctly.
5. Use Wireshark to view what is actually happening on the Ethernet port.
6. Contact Curtiss-Wright support (acra-support@curtisswright.com) for further support.

This page is intentionally blank

29 Dec. 2017 | TEC/NOT/058