Jünger Audio-Studiotechnik GmbH
Manfred Ottenbreit

Structure of Junger Audio SNMP management information

Changes from JUNGER-C8k-MIB-1 to Junger-C8k-MIB-2:
The value **syncMasterNotInstalled(3)** is added to the syncMasterStatus
The OIDs: **fanOneSpeed**, **fanTwoSpeed**, **fanStatus** have been added
The Trap: **fanFail (8)** has been added
The list of **moduleType** has been changed
The modules C8082, C8088, C8601, C8611, C8612 have been added

Changes from JUNGER-C8k-MIB-2 to Junger-C8k-MIB-3:
The variables of the procStatus have been renamed:
procInRange(1) from gainInRange(1)
procOutOfRange(2) from GainOutOfRange(2)
The Trap: procError (7) has been renamed (from procStuck)

Changes from JUNGER-C8k-MIB-3 to Junger-C8k-MIB-4:
Variable bindings removed from the Trap objects
procModules table removed
Monitoring of :
- C8080 removed from the agent
- C8081 removed from the agent
- C8446 removed from the agent
- C8405 added to the agent
- C8491 added to the agent

Trademark "Level Magic" exchanged by "Level Magic™ II"

Important note! The comments of the MIB-4 regarding supported modules are not 100% consistent with this SMI description. We will make a change to the comments soon, without changing the MIB version since this is not relevant for the functionality.

1. Introduction

The structure of management information is based on SMIv1.

Beside the enterprise specific information, the SNMP agent will support **MIB-II** for the following object groups:
system Group
interface Group
ip Group
icmp Group
tcp Group
udp Group
snmp Group

From the protocol point of view, we will support the **SNMPv1** based **„GetRequest“**, „**GetNextRequest**“, **“GetResponse”** and „**Trap**“ PDUs only. We do **not** allow for “**SetRequest**” PDU.

Defined objects which are not supported by the agent will return the error code **“noSuchName”**

The enterprise specific information of the C8000 system is defined in the **JUNGER-C8K-MIB-x.mib** file**.**

2. Description of the system Group

2.1 sysDescr.0 [octet string]

C8702, Revision 4, Image rel_1_14_00_15564/Red Hat eCos 2.0.40/ucd-snmp-4.1.2

“C8702”	- Module name of the c8k Frame Controller
“Revision x”	- Hardware revision of that module
“Image rel_x_y_z abcd”	- firmware version and image [abcd] number of the frame controller
“Red Hat eCos x.y.zz”	- version of the **eCos** operating system, licensed from **eCosCentric**
“ucd-snmp-x.y.z”	- SNMP integration of the UC Davis, used by the Red Hat for the eCos OS.

2.2 sysObjectID [object identifier]

This object identifier points to the first device specific object in the enterprise tree. In case of the C8000 frame it has the value: {enterprises.25999.3.1.1} to allow the manager to find the entry for device specific information

2.3 sysUpTime [timeticks]

The uptime of the c8k frame from last cold start or warm start counted in: time ticks of 100ms:
days, hours, minutes, seconds and 100ms system time ticks

2.4 sysContact [octet string]

The management contact e-mail address. This address can be set by use of the web interface.

2.4 sysName [octet string]

the given frame name for the c8k frame. This name can be set by use of the web interface. It should identify the frame by description of its main purpose

2.5 sysLocation [octet string]

The given location name of the c8k frame within the technical infrastructure of an organization / technical room etc.

2.6 sysServices [INTEGER]

See RFC 1213 to calculate the number. We define our frame system as an IP based host that provides an application so the value is set to $72=2^{(4-1)} + 2^{(7-1)}$.

3. The interface Group

3.1 ifNumber.0 [INTEGER]
The number of data interfaces implemented (3 in this case)

The interface group is represented in a **table** form:
3.2.1 ifTable
The start OID for that table. This object is not-accessible because it is only known per definition by the MIB.
3.2.2 ifEntry
The table index that defines the columnar objects of that table. This object is not-accessible because it is only known by the MIB. The columnar objects are:
3.3.1 ifIndex [INTEGER]
A unique number for each interface (it represents the rows within the if table). It is used as a reference by the interface group itself and the interface specific information such as IP input packets (if applicable) for such interface
3.3.2 ifDescr [OCTET STRING]
A textual description of the referred interface (e.g Motorola MCF52xx Ethernet device)
3.3.3 ifType [integer]
A textual description of the type of interface (e.g. softwareLoop back).
3.3.4 ifMtu [INTEGER]
Defines the largest size in octets of a PDU that can be sent/received on that interface
3.3.5 ifSpeed [gauge]
The data rate of the interface in bits/sec
3.3.6 ifPhysAddress [OCTET STRING]
The physical address of the interface, if applicable (e.g. MAC address)

and so forth …..

See **RFC1213** for more details also for the **groups** listed below.

4. The ip Group

5. The icmp Group

6. The tcp Group

7. The udp Group

8. The snmp Group

9. The enterprise specific objects

The device specific objects of the C8k system as part of the enterprise tree are divided into frame and module specific objects. The frame specific objects are simple scalar objects while the module objects are organized as table objects.

9.1 **frame -** specific objects

9.1.1 temperature [INTEGER]
It reads the absolute temperature in degrees Celsius with one degree resolution, measured on the surface of the C8702 frame controller.

9.1.2 powerStatus [INTEGER]
The 2ower status represents the state of the double power supply for the frame. The normal value is:
powerGood(1)
If the load balancing is out of order or if one of the two power supplies breaks, the value will be set to:
powerFail(2)

9.1.3 syncMasterStatus [INTEGER]

The sync master status represents the state of the master sync module C8820 / C8830. The normal value is:
syncMasterLock(1)
If the external sync disappears or if the master module breaks, the value will be set to:
syncMasterUnlock(2)
If no sync master module is installed in case of C8934 split frame or C8942 compact frame, the value will be set to: **syncMasterNotInstalled(3)**

9.1.4 syncSlaveStatus [INTEGER]

In case of two sync modules are installed for fail safe operation the normal value is:
syncSlaveLock(1)
If the external sync disappears or if the slave module breaks, the value will be set to:
syncSlaveUnlock(2)
If no sync slave module is installed, the value will be set to:
syncSlaveNotInstalled(3)

9.1.5 modulesStatus [INTEGER]

The modules status represents the summary state of all manageable modules of a frame. The normal value is:
modulesGood(1)
If one module loses communication with the frame controller the value will be set to:
modulesFail(2)

9.1.6 temperatureStatus [INTEGER]

The temperature status represents the state of the temperature measured on the surface of the frame controller C8702 module. The normal value is:
temperatureLow(1)
If the temperature reaches 50degress Celsius, the value will be set to:
temperatureHigh(2)
If such a situation occurs and if the temperature falls below 45degrees Celsius,
the value will be reset to:
temperatureLow(1)

9.1.7 interfaceStatus [INTEGER]

The module controller provides the input status of a multi channel input module (SDI, MADI, Ethersound) or the summary status of all AES/EBU inputs of an AES/EBU input module. The frame controller summarizes such information for all modules. The normal state is:
interfaceInputGood(1)
If an error condition is detected at one of the above mentioned inputs the state changes to:
interfaceInputLost(2)

9.1.8 busStatus [INTEGER]

The module controller summarizes the status of all relevant audio bus inputs of either processing modules or output modules which receive their audio data from the frame bus. This information is again summarized by the frame controller for all of the modules installed. The normal state is:
busInputGood(1)
If an error condition is detected at one of the above mentioned inputs the state changes to:
busInputError(2)

9.1.9 procStatus [INTEGER]

The summary status of all processors (DSP modules) installed in a frame. The DSP generates a weighted value of the momentary gain of each processing channel (see also procModules). The module controller “looks” if such value exceeds a given threshold for a certain amount of time for a specific processing channel and it summarizes the status for all processing channels. This information is again summarized by the frame controller for all of the processing modules installed. The normal state is:
procInRange(1)
If this gain resides above a predefined threshold for a certain time for any processing channel, the state changes to:
procOutOfRange(2)

9.1.10 fanStatus [INTERGER]

If the revolution per minutes of one of the two fans is below 40r.p.m. the value will be set to:
fanSpeedLow(2)
If the revolution of the fan is back above 60r.p.m. the value will be set back to:
f**anSpeedNormal(1)**

9.1.11 fanOneSpeed [INTEGER]

It reads the revolution per minute of the first fan of a C8942 frame with one r.p.m. resolution

9.1.12 fanTwoSpeed [INTEGER]

It reads the revolution per minute of the second fan of a C8942 frame with one r.p.m. resolution

9.2 **modules** specific objects (detailed module status information)

Depending on the type of module the controller can detect if the physical input is lost (e.g. the AES or SDI carrier is not present) or if there is a bus error. Therefore the bus driving entity generates the error detection flag(s). Both errors are of high severity because it is an indication that no or corrupted audio data is present. This information is collected by the modules table. This table contains all manageable modules. Since Junger Audio is on a migration path, there may be legacy “dumb” modules in a frame which do not have a module controller to communicate with the frame controller. Such modules can not be monitored directly. As long as such modules generate error flag(s), they can be implicitly monitored by the next module in the chain.

9.2.1 modulesNumber [INTEGER]

The number of manageable modules installed within a frame. These are modules controlled by a module controller.

9.2.2 modulesTable [SEQUENCE OF ModulesEntry]

This table organizes the objects to watch the system on the module level. The agent will create such table depending on the number of modules found on the CAN bus after power up. If a module is removed or dies while in operation the table row will still be maintained by the agent. It will be further used if the module appears again (exchanged or self repaired ;-) with the same ID (CAN bus address). If a new module is added with a different CAN bus address, the agent will create a new row for that table and will increase the **modulesNumber** and the **modulesIndex** accordingly.

Important note! The CAN ID is used to place a module icon for the web interface modules overview.
It is recommended and common practice to select such addresses according to the mechanical place within the frame. The C8000 system supports 32 addresses for manageable modules.
The Frame Controller and the Sync Interface(s) have unique addresses which can not be changed by the customer. They do not appear in the **modulesTable**.

9.2.3 modulesEntry [ModulesEntry]

9.2.3.1 moduleIndex [INTEGER]

A Unique number for each module within a frame. Its value ranges between 1 and the value of **modulesNumber**. The range of the value is 1 ... 32 (the maximum number of modules due to the CAN address scheme).

9.2.3.2 moduleID [INTEGER]

CAN address of the module (setting of the CAN address selector). The range of the value is 0x0 … 0x1F

9.2.3.3 moduleName [OCTET STRING]

A16 character name that is stored by the module

9.2.3.4 moduleTyp [OCTET STRING]

The module type is represented by a 5 character module type expression:

C8046	-- 4CH Level Magic™ II processor
C8082	-- 5.1 Failover / Switch Over / Ducking
C8086	-- 8CH Level Magic™ II processor
C8087	-- 5.1 Upmix processor
C8088	-- 8CH Mix Matrix with Limiter
C8188	-- 8 AES I/O Sub-D
C8189	-- 8 AES I/O BNC
C8305	-- MADI BNC I/O
C8306	-- MADI dual optical I/O
C8402	-- 8CH SD SDI Embedder/De-Embedder
C8403	-- 8CH HD SDI Embedder/De-Embedder
C8404	-- 16CH HD/SD-SDI Embedder/De-Embedder with Video Delay
C8405	-- 16CH 3G/HD/SD-SDI Embedder/De-Embedder with Video Delay
C8486	-- 4/8 CH Level Magic™ II + HD/SD-SDI Processor [SDI-DSP]
C8491	-- 4/8/16 CH Level Magic™ II + 3G/HD/SD-SDI Processor
C8502	-- reserved (Monitor Module new)
C8601	-- Dolby-E & AC-3 Decoder
C8611	-- Dolby-E Encoder
C8612	-- Dolby Digital (AC-3), Dolby Digital plus (E-AC-3) Encoder
C8651	-- Dolby Metadata (serial) I/O
C8685	-- Audio Delay
C8817	-- 8 Isolated GPI/Os with balanced Aux Voltage

9.2.3.5 interfaceInputStatus [INTEGER]

Status of the physical input of a multi channel input module (SDI, MADI) or a multi input module (AES/EBU), defined by 8 bits encoded to an INTEGER, where the bit position Bx is bound to:

inputMultichannelGood	B7=0	-- status of a multi channel input (SDI, MADI, Ethersound)
inputMultichannelError	B7=1	--
	B6=x	-- not defined
	B5=x	-- not defined
	B4=x	-- not defined
inputFourGood	B3=0	-- the status of a physical AES/EBU input #4
inputFourError	B3=1	--
inputThreeGood	B2=0	-- the status of a physical AES/EBU input #3
inputThreeError	B2=1	--
inputTwoGood	B1=0	-- the status of a physical AES/EBU input #2
inputTwoError	B1=1	--
inputOneGood	B0=0	-- the status of a physical AES/EBU input #1
inputOneError	B0=1	--

9.2.3.6 busInputStatus [INTEGER]

Status of the bus inputs of a processing module or the bus inputs of a physical output module (SDI, AES/EBU, MADI). It is defined by 32 bits, encoded to an INTEGER. Each bit represents the status of a bus line. The number of the bus line (32 to 1) is given by its bit “position” (31 to 0) in Big Endian format.

No error detected for bus line x	Bx=0
Error detected for bus line x	Bx=1

9.2.3.7 moduleProcStatus [INTEGER]

Status of the processing of a logical channel (maximum is 8 channels), defined by 8 bits encoded to an INTEGER where the bit position Bx represents one processing channel. The main parameter of a processing channel is the development of the system gain over time. E.g. if the leveller gain is below (negative gain) or above (positive gain) a threshold, defined by the **Leveler Max Gain** for more than 10 sec, the **processingXBad** condition will be set. If there is no processing channel (e.g. channel 6 of a 4 channel module) the values are set to **Bx=0**. The **moduleProcStatus** object is not valid for I/O modules at all. In this case all of the 8 bits are not valid and will be set to Bx=0.

processingXGood	Bx=0
processingXBad	Bx=1

Because the limiter gain change of the Level Magic process will be compensated by the Transient Processor for mid term gain changes and by the Leveler Process for long term gain changes, the observation of the **Limiter Gain Reduction** makes not much sense.

The **moduleProcStatus** OID is supported only by the C8086, C8086 (-M), C8086+ (-M), C8446, C8486, C8486-8 and 8491. The other Level Magic products, C8007 and C8046 do not support this management information.

The portfolio of Junger Audio products includes modules which are not **Level Magic** processors but have other processing features. Junger Audio supports the monitoring of the limiters of the **C8088 Mix Matrix.**
The momentary gain reduction value is low pass weighted (τ=3sec.) for monitoring purposes. I.e. a jump of 6dB will take about 24sec to reach the monitoring end value. If this weighted **Limiter Gain Reduction** exceeds **-6dBFS**, the **limiterXBad** condition will be indicated. If the **Limiter Gain Reduction** returns above **-6dB** the **limiterXBad** status will be cleared.

limiterXGood	Bx=0
limiterXBad	Bx=1

Another case is the **C8082 5.1 Failover / Switch Over / Ducking** module. It provides **6 limiters** for the mixing nodes. But it also detects **fail conditions** of its audio inputs. Therefore the following conditions will be indicated by the 8 bits Bx.

limiterXGood	Bx=0
limiterXBad	Bx=1
No Failure detected for 5.1 or main stereo	B6=0
Failure detected for 5.1 or main stereo	B6=1
No Failure detected for AUX stereo	B7=0
Failure detected for AUX stereo	B7=1

The introduction of the Dolby® processing modules 8601, 8611, 8612 lead to an extended interpretation of the **moduleProcStatus** OIDs:

Metadata valid	B1=0	
Metadata not valid	B1=1	
Frame Reference OK	B2=0	C8611 only
Frame Reference failure	B2=1	C8611 only
Status Encoder 1 Good	B3=0	
Status Encoder 1 Bad	B3=1	
Status Encoder 2 Good	B4=0	C8612 only
Status Encoder 2 Bad	B4=1	C8612 only
No Decoding Error	B5=0	C8601 only
Decoding Error	B5=1	C8601 only

9.3 **procModules** (QoS parameter)

The DSP of the digital dynamics processor generates a weighted signal that represents a moving average of the summary gain of the **Level Magic** process for each processing channel. This signal is low-pass filtered using a time constant of approx. 1min. The SNMP manager or a dedicated application may poll this variable to display a plot or create a table or put the data into a data base for later evaluation. A recommended polling rate is 5 to 10 times per second. The unit is dB and the resolution is 0,2dB.

9.3.1 procModulesNumber [INTEGER]

The number of processing modules within a frame

9.3.2 procModulesTable [SEQUENCE OF ProcModulesEntry]

This table organizes the objects to watch the system on the processing (DSP-) module level. The agent will create such table depending on the number of processing modules found on the CAN bus after power up. If a module is removed or dies while in operation the table row will still be maintained by the agent. It will be further used if the module appears again (exchanged or self repaired ;-) with the same ID (CAN bus address). If a new module is added with a different CAN bus address, the agent will create a new row for that table and will maintain the **procModulesNumber** and **procModulesIndex** accordingly.

To reduce the amount of traffic inside the c8k frame, the value will be gathered by the frame controller on request of the SNMP manager. Therefore, it is not possible to use the **variable bindings** field of the “GetRequest” PDU. The manager must poll each **procModuleChannel** OID by “GetRequest” or “GetNextRequest”.

9.3.3 procModulesEntry [ProcModulesEntry]

9.3.3.1 procModuleIndex [INTEGER]

A unique number for each processing module of a frame. Its value ranges between 1 and the value of procModulesNumber.
The range of the value is 1 ... 32 (the maximum number of modules due to the CAN address scheme)

9.3.3.2 procModuleID [INTEGER]

CAN address of the processing module (setting of the CAN address selector). The range of the value is 0 …. 15

9.3.3.3 procModuleName [OCTET STRING]

A 16 character name that is stored by the module

9.3.3.4 procModuleTyp [OCTED STRING]

The module type is represented by a 5 or 6 character module type expression

C8007	-- 2CH FM Broadcast Processor + Level Magic
C8086	-- 8CH Level Magic Processor
C8088	-- 8CH Mix Matrix with Limiter
C8446	-- 4CH Level Magic SD-SDI
C8486	-- 4/8CH Level Magic HD/SD-SDI [SDI-DSP]

9.3.3.5 procCh”X” [INTEGER]

The value represents the weighted gain in dB with 0,2dB resolution.

9.3.3.5.1 procChOne [INTEGER]

The first processing channel of a processing module

9.3.3.5.2 procChTwo [INTEGER]

The second processing channel of a processing module

And so forth …

9.3.3.5.8 procChEight [INTEGER]

The eighth processing channel of a processing module

10 The trap objects

If a situation described by the trap name below occurs, the agent will send a SNMPv1 trap number to the specified "TrapSinkAddress". In case of trap numbers 5, 6, and 7 the manager must poll the "modulesTable" of the frame for details afterwards.

10.1 Generic SNMPv1 traps

TrapName	TrapNumber
coldStart	0
warmStart	1
linkDown	*2*
linkUp	*3*
authentication-Failure	4

We have extended the meaning of this trap. Beside the attempt to get OID values by using the wrong community string, we will also send this trap number if access control for the web interface is enabled and somebody tries an unauthorised log in.

10.2 Enterprise specific Traps

TrapName		TrapNumber
temperatureHigh		1
Send trap (1)	if temperatureStatus=tempHigh(2)	
powerFail		2
Send trap (2)	if powerStatus=powerFail(2)	
syncFail		3
Send trap (3)	if syncMasterStatus=syncMasterUnlock(2)	
or	if syncSlaveStatus=syncSlaveUnlock(2)	
modulesFail		4
Send trap (4)	if modulesStatus=modulesFail(2)	
inputLost		5
Send trap (5)	if interfaceStatus=interfaceInputLost(2)	
busError		6
Send trap (6)	if busStatus=busInputError(2)	
procError		7
Send trap (7)	if procStatus=procOutOfRange(2)	
fanFail		8
Send trap (8)	if fanStatus=fanSpeedLow(2)	