MediaMatrix®

A Division of Peavey Electronics Corporation

NWare

External control user guide

Version 1.4.2.0

May 12, 2008

Prepared by Peavey Digital Research, 6 Elm Place, Eynsham, Oxford, OX29 4BD, UK.

Email:mmservice@peavey-eu.com.

Contents

Chapter 1

About this guide

In This Chapter

Scope

This guide describes how to use different protocols, such as PASHA and SNMP, to remotely control and monitor devices in an NWare project.

Documentation conventions

The following are used in the documentation to highlight particular sections of information.

Tip: Suggests alternative ways of completing a task and shortcuts that might not otherwise be obvious.

Note: Indicates important information that should not be ignored.

Caution: Indicates that unless you are careful, your actions could result in equipment damage or loss of data.

Warning: Indicates that unless you are careful, your actions could result in injuries to personnel.

Manual set

This guide is part of the MediaMatrix documentation set.

The table below shows which user guides to refer to when you want to find out how to accomplish various tasks.

Tasks	**Relevant Guides**
Creating and managing projects using NWare	NWare User Guide
Using devices available from the NWare device tree.	NWare Device Reference
Finding out about new features added to releases of NWare and NION software	NWare Release Notes
Using different protocols, such as PASHA and SNMP, to remotely control and monitor devices in an NWare project.	External Control User Guide

Tasks	**Relevant Guides**
Physical installation and initial configuration of a NION digital audio processor.	NION Hardware Documentation
Understanding the features and physical characteristics of the NION digital audio processor.	NION Hardware Documentation
Physical installation and initial configuration of a Cab4n CobraNet audio bridge.	Cab4n Manual
Physical installation of a ControlManager server. Installation and configuration of associated software.	Control Manager Primer
Understanding how Pandad works and using it on your network.	Pandad Primer

Chapter 2

Introduction

In This Chapter

Remote Control

The NION platform is totally built around remote control. NWare itself connects as remote control software (mostly via TCP/IP) to one or dozens of NioNodes. In this topic, we'll refer only to NioNodes (individual NION units), but you should know that ControlNodes (generic term for PC's running the Control Manager application -- a.k.a. ConMan) also support these protocols.

There are several methods of remote control of NioNodes. There are two categories of remote control: Native and External.

Native Control is what you use to set up and control NioNodes directly. In other words, you are using programs specifically designed to configure and control NioNodes.

External Control comprises protocols that you can use to manipulate settings from other control systems or panels.

Native Control

NWare : Design

NWare : Design (or just NWare) is the primary software that you use to design NION projects. Not generally thought of as Remote Control, logging, status, deployment, scripting and all other functions that NioNodes are to perform are configured with NWare.

For more information, refer to the NWare User Guide.

NWare : Kiosk

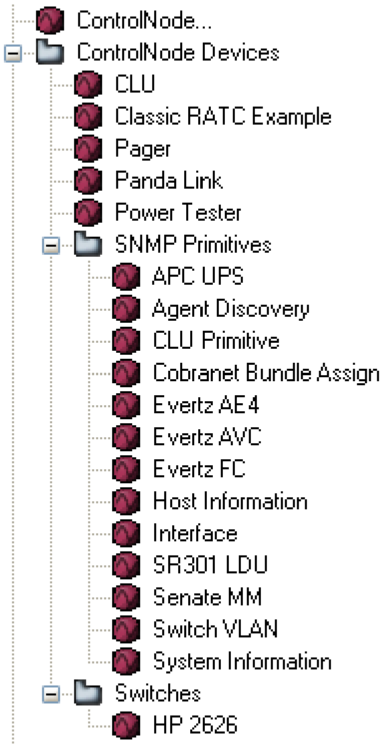
NWare:Kiosk or just *Kiosk*, is a control only GUI, which is launched by running the NWare application with the *personality* command line argument. Kiosk allows the user access to a design created in NWare, but not to change its configuration. Furthermore, the UI features of NWare may be tailored by the designer, in order to restrict specific user actions as required.

For more information, see Running NWare in Kiosk mode in the NWare User Guide.

NWare++

NWare++ contains all the functionality of NWare, but adds extra devices specifically for using the SNMP features of Control Manager (also known as *ConMan*). Since NioNodes do not have the ability to read and write SNMP values, these tools are only useful for designing projects for ControlNodes. Also, there is a ControlNode device for assigning devices to a specific ConMan. It is used like the NioNode device is used for NIONs.

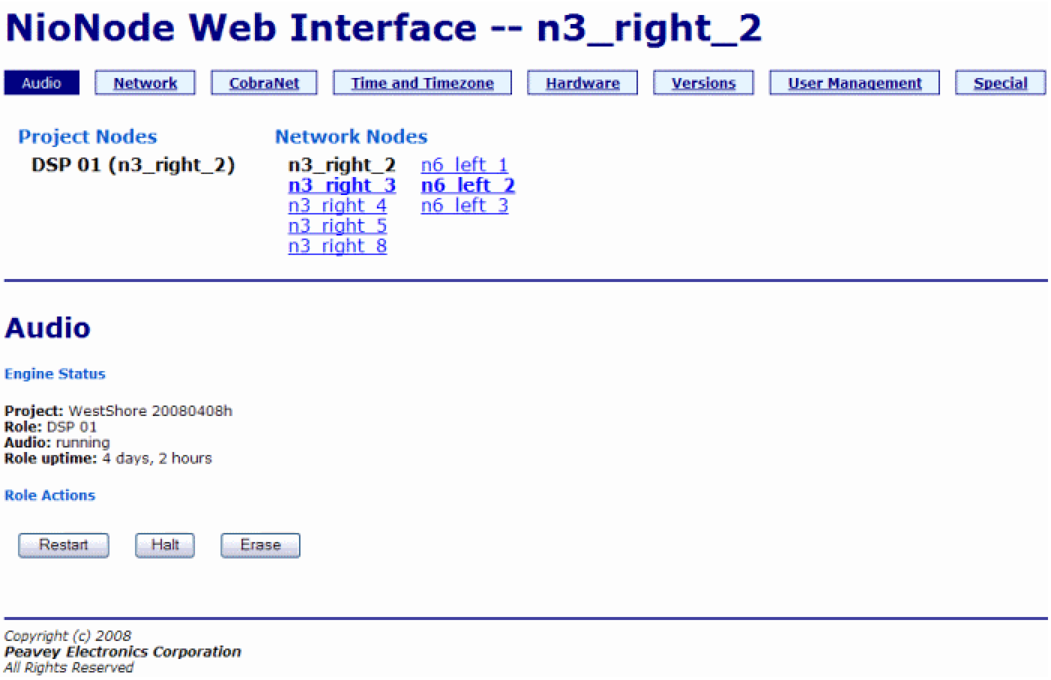
Tip: The NWare++ tree lists all the devices you can control using SNMP. If you cannot find a particular device, we may be able to provide you with a custom built ControlNode device, as long as the uncompiled MIB file can be provided. The device can then be made available to other users in a future release of NWare.



NioNode Web Interface

Each NION has a built-in web server for individual unit status, versions, debugging, box-level account management, etc. This also allows you to easily assign unit time, time zone, name, IP settings, and more.

Note: Some features can and should be restricted by changing the 'defaultuser' account permissions from the web interface's User Management section.



NioNode Web Interface -- n3_right_2

Project Nodes

DSP 01 (n3_right_2)

Network Nodes

n3_right_2
n3_right_3
n3_right_4
n3_right_5
n3_right_8
n6_left_1
n6_left_2
n6_left_3

Audio

Engine Status

Project: WestShore 20080408h
Role: DSP 01
Audio: running
Role uptime: 4 days, 2 hours

Role Actions

Copyright (c) 2008
Peavey Electronics Corporation
All Rights Reserved

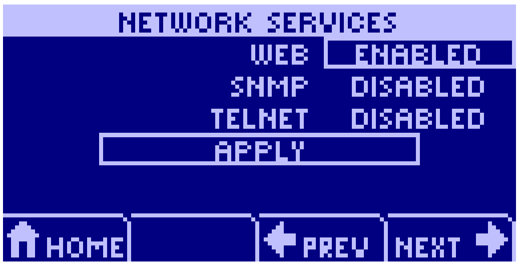
Tip: By clicking **Special**, and then **Advanced** from the screen shown above, you can view the screen that is currently shown on the NION front panel. The image can then be copied and pasted into your documentation and emails.

Enabling or disabling the web interface

The web interface is disabled by default on NioNodes and must be enabled before you can access a NION via a web browser.

⏩ *To enable or disable the web interface*

1. From the main menu, select **CONFIG** to display the first configuration page, **LAN CONFIG**, then select **NEXT** repeatedly until the **NETWORK SERVICES** page is displayed.



NETWORK SERVICES

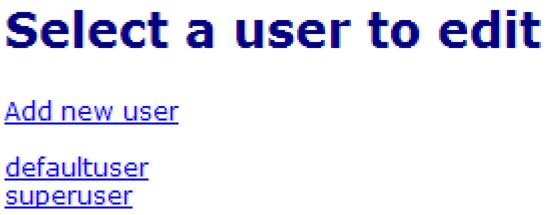
WEB ENABLED
SNMP DISABLED
TELNET DISABLED

2. Use the wheel and the wheel push button to change the **WEB** setting to **ENABLED** or **DISABLED**.
3. Select **APPLY**.

When the web interface is enabled, you can enter the IP address of the unit into your web browser to get to the web interface.

Tip: If this does not work, you may have an IP address problem on either the NioNode or your PC. If you have a proxy server set up in your internet options, you may have to create an exception for local IP addresses.

User Manager



Select a user to edit

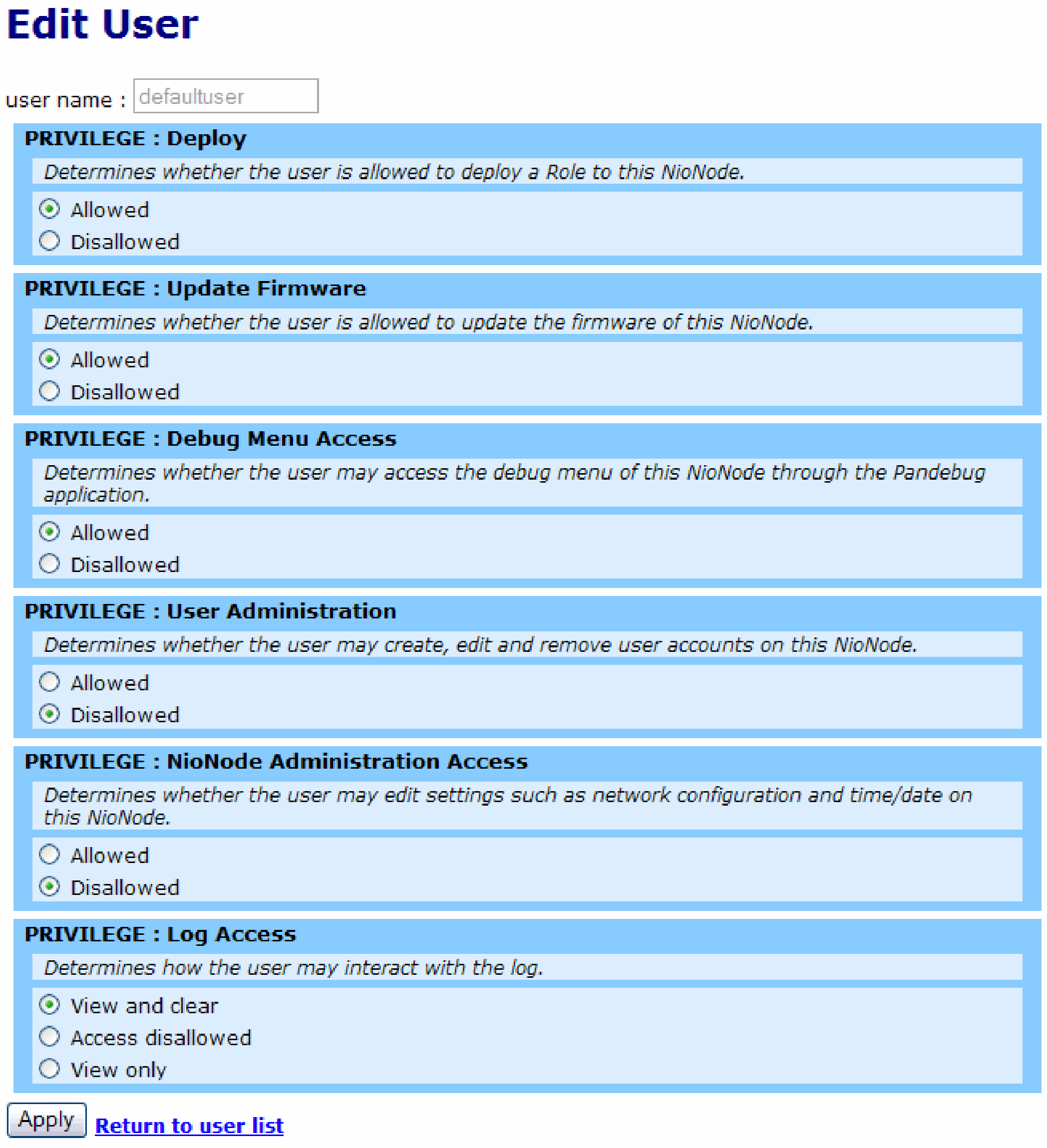
Add new user

defaultuser
superuser

User Manager allows you to set up accounts on the specific NioNode unit. You can choose to restrict or enable specific users from deploying projects, updating firmware, entering the user manager, viewing the log files, change network settings, halting or restarting NioNodes or the PIOND program that runs on that unit, etc.

The defaultuser account represents the features that are available to all users without supplying a username and password. We strongly encourage you to disable as many functions from the defaultuser as possible upon completion of an installation, or even before completion. defaultuser cannot be deleted.

The superuser account always has all permissions and should be used by the installer or main administrator of the system. The Superuser account defaults to having no password, but one should be chosen soon after configuring the unit, since the default will soon become common knowledge amongst SI's. superuser cannot be deleted.



Edit User

user name : defaultuser

PRIVILEGE : Deploy

Determines whether the user is allowed to deploy a Role to this NioNode.

● Allowed
○ Disallowed

PRIVILEGE : Update Firmware

Determines whether the user is allowed to update the firmware of this NioNode.

● Allowed
○ Disallowed

PRIVILEGE : Debug Menu Access

Determines whether the user may access the debug menu of this NioNode through the Pandebug application.

● Allowed
○ Disallowed

PRIVILEGE : User Administration

Determines whether the user may create, edit and remove user accounts on this NioNode.

○ Allowed
● Disallowed

PRIVILEGE : NioNode Administration Access

Determines whether the user may edit settings such as network configuration and time/date on this NioNode.

○ Allowed
● Disallowed

PRIVILEGE : Log Access

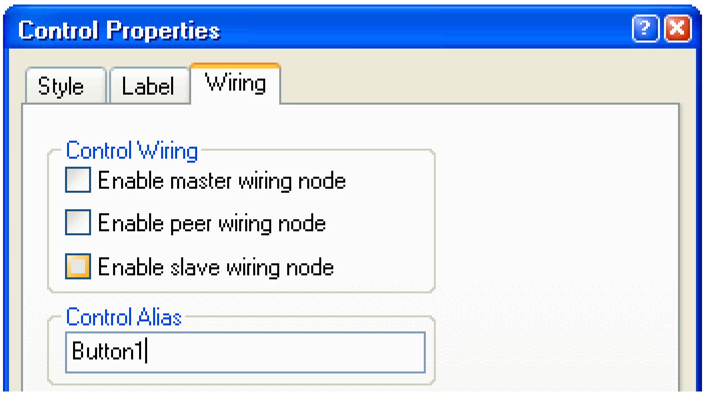
Determines how the user may interact with the log.

● View and clear
○ Access disallowed
○ View only

External Control

Controlling Controls

During runtime, those Controls that have been given a Control Alias by the designer can be read and written by external control systems via a TCP/IP or serial connection. The external control system need connect to only one NioNode or ControlNode in the Project. It is not necessary to know which Node a given Control actually resides on as all Controls with Control Aliases are available on all Nodes. For redundancy, though, a control system designer could arrange for failover to alternate Nodes if the default one it connects to stops responding.



☐ Enable master wiring node
☐ Enable peer wiring node
☐ Enable slave wiring node

Button1

Three protocols are supported: RATC v1, RATC v2 and PASHA. Each of these protocols is available either through a TCP/IP (network) connection, and/or through a serial connection. RATC protocols have never before been available via a serial interface.

Configuring

The TCP/IP and Serial control services are configured in the NioNode or ControlNode device.

Testing External Control with the Emulator

It is possible, and helpful, to verify and test TCP/IP and serial external control using the NWare Emulator. The thing to keep in mind is that, since the Emulator emulates all NioNodes and ControlNodes on the NWare PC, you can have port conflict errors if your Project has external control enabled on more than one Node. You can, of course, change the ports on the Nodes so that they don't conflict when in use together on the Emulator. This applies to both TCP/IP ports and serial (COM) ports.

RATC1 and RATC2

RATC, or Remote Access Terminal Control, is a set of command-line based protocols that allows a remote client program to set and get the control values in a compiled and/or deployed project file. The remote client communicates with MediaMatrix via a TCP network connection, so RATC is compatible with both local area networks and the Internet. Any number of RATC clients (within reason) may connect simultaneously to a NION. The RATC1/RATC2 services are configured and enabled through the NioNode device within NWare.

RATC1 is the first generation protocol that was used in Classic frame-based MediaMatrix systems. RATC1 in NION systems is equivalent to what was called RATC in Classic MediaMatrix.

RATC2 is the improved version of RATC that was introduced with the NION platform. Very similar to RATC1, RATC2 introduces shortened commands, and a few extra functions. One of the most exciting change for third-party programmers is the Change Group Scheduling feature, which can be configured to automatically send changed values without being polled.

Although it is possible to use a TELNET style application to control MediaMatrix with RATC, custom software is probably required for real world applications. TELNET is handy for testing your RATC connection.

PASHA

PASHA, the MediaMatrix Serial Handling Adapter, is a remote control protocol that provides external serial command and read-back of any of the controls appearing in a MediaMatrix view file.

PASHA is a lower level and lower performance control than RATC1 or RATC2. It is primarily intended to be driven by programmable serial control devices.

PASHA is only available from the front RS-232 port or the rear RS-422/RS-485 port.

Chapter 3

PASHA

In This Chapter

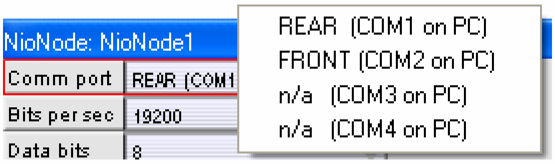
Introduction

PASHA, the MediaMatrix Serial Handling Adapter, is a remote control protocol that provides external serial command and read-back of designer selected controls in a NION project. PASHA is only available via the front RS-232 or the back RS-422/RS-485 serial ports.

The NION version of PASHA does not implement the full protocol as exists in Classic MediaMatrix. The NION version does not support PASHA Change Groups, mainly because the RATC protocols are available from the serial ports and offer enhanced Change Group functionality.

Only one serial port and protocol are allowed on each NioNode in a project.

The serial command protocol used by PASHA is human-readable ASCII, so it is possible to test and debug PASHA control of a specific project using just a Windows-based PC with a spare serial port - NION hardware is not required. When emulating a NioNode using NWare on a PC, the PC's serial ports can simulate the PASHA behavior of a NION as seen in the image below.



NioNode: NioNode1

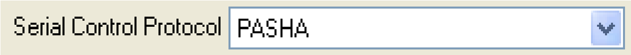
Comm port: REAR (COM1

Bits per sec: 19200

Data bits: 8

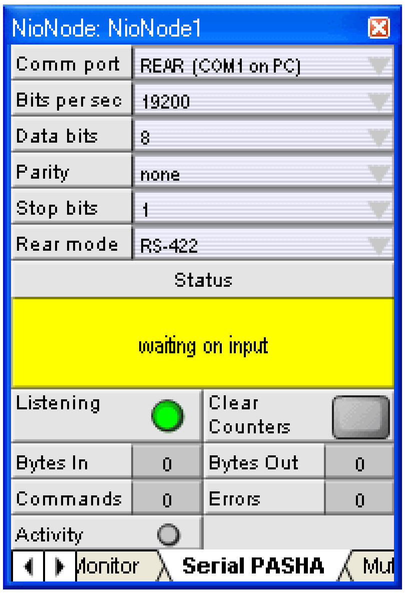
Configuring PASHA

PASHA is configured in the NioNode device within NWare. PASHA is an option in the Serial Control Option of each NioNode device.



Serial Control Protocol: PASHA

Once the NioNode device has been created or modified with the PASHA option, there will be a Serial PASHA tab inside the block.



NioNode: NioNode1

Comm port: REAR (COM1 on PC)

Bits per sec: 19200

Data bits: 8

Parity: none

Stop bits: 1

Rear mode: RS-422

Status: waiting on input

Listening ●

Bytes In: 0

Bytes Out: 0

Commands: 0

Errors: 0

Activity ○

The project must be deployed or emulated in order to see and change the settings you see here.

User IDs and Control Groups

PASHA user IDs correspond to 3 character Control Group names in View Files. To specify a user ID for a control in MediaMatrix, open its Control Object Properties page and select the Group tab. Enable Control Grouping and give the control the desired 3 character Group Name. Any alphanumeric character may be used, and the protocol is case sensitive, so "aaa" is different from "AAA". Control Group names of other lengths are possible in MediaMatrix, but only those exactly 3 characters long can be used for serial control via PASHA.

Message Protocol

Messages are ASCII (text) strings. While they contain hexadecimal numbers, the Hex numbers are represented by ASCII (text). Do not send actual hex data, but an ASCII (text) string that represents the value to be sent.

Each message begins with a message-type character and ends (with the exception of the NAK message) with an end-of-message character '.'. Each message to PASHA results in a response message from PASHA. PASHA never sends a message except in response to an external message.

In general, the protocol supports Setting and Getting control values. User IDs are entered into Control Groups which identify a unique control or group of controls. They are specified for use with PASHA with 3 alphanumeric characters. Control values are specified with a 2 digit hexadecimal number, 00 through FF.

Controls are set to a position using the cSETVALUE 'S' command. You can determine the current setting of a control using the cGETVALUE 'G' command. The 256 values that a control can take on correspond to 256 equidistant positions of a control in a MediaMatrix view file. It is possible, and normal in some cases, for the value that a control actually takes on (and returns) to not match the value to which it was set. For instance, in the case of a switch-style control, any value sent between 00 and 7F will result in a return value of 00, and any value sent between 80 and FF will result in a return value of FF. This is because a switch can only be either ON or OFF. If it is ON, it will return a value of FF, while if it is OFF, it will return a value of 00. If commanding a Level with Trim device's Trim control, the dB range of the Trim control (set up by Trim Min. and Trim Max.) is simply divided into 256 possible values. Most devices' gain controls, however, are non-linear, and the math is not so simple. For those controls, tables are included below that map linear control values to dBs of gain, one for a gain control with range of -100dB to +12dB, another for a gain control with a range of -100dB to 0dB. Also included are tables that map control values to Router channel selections for various sized Routers.

For example, to set the channel 2 input gain specified in the demo view file PASHA32.PAV to fully counter clockwise, one first look up the UID specified in the Control Group for the control in question. In this case it is 2iG. Fully counter clockwise is 00 Hex. The cSETVALUE command is 'S' and the end-of-message character is '.'. Putting this all together gives a command string:

```
S2iG00.
```

Note: It is not required or desirable to put a carriage return after the end-of-message (EOM) character '.'. PASHA will accept the command when it receives the EOM character. Shortly thereafter it will return the value that the control has been set to:

`V2iG00.`

To check what a control is set to, you can use the cGETVALUE 'G' command. For example to check the setting of the channel 1 parametric EQ center frequency control, you send:

`G1Cf.`

Note: Since you are requesting a control value, you do not send one. If the control was set to the midpoint of its rotation you will get a response:

`V1Cf80.`

Remember it is possible, and normal in some cases, for the value that a control actually takes on (and returns) to not match the value to which it was set. For instance, in the case of a switch-style control, any value sent between 00 and 7F Hex will result in a return value of 00 Hex, and any value sent between 80 and FF Hex will result in a return value of FF Hex. For example you may have sent to the Channel 2 output mute:

`S2oM3E.`

However since the switch can only be on or off it will be set to off and return:

`V2oM00.`

If you send a command using a UID that is not in a Control Group in the view file, PASHA will respond with a cUNLISTEDUID 'U' and repeat back the Uid it does not find in the PASHA.INI file. For example if there was no Control Group hat in the view file, and you sent:

`Shat48.`

the response would be:

`Uhat.`

If you send a command using a valid UID (a 3 character Control Group name) in the view file, but one or more of the controls that Control Group references is not found in the view file, PASHA will respond with a cUNLISTEDCID 'C' and repeat back the UID you sent. For example if there was a UID ukc in the view file, and one or more of the controls to which it referred are "dead" or "orphaned", and you sent:

`Sukc48.`

the response would be:

`Cukc.`

A "dead" control will occur if the algorithm it is part of did not compile. This will happen if the algorithm is not properly wired. An "orphan" control will occur when the algorithm it was copied from has been deleted from the view file.

If you send an illegal message such as specifying a value as part of a get message, or scramble the order of message components, or there are certain other communications errors, PASHA may respond with a Negative Acknowledge mNAK '?' instead of the normal response. For example if you sent:

`GHLP57.`

PASHA will respond with:

`?`

Note: This is the only message in the PASHA protocol, which does not end in an end of message character cEOM '.'.

No response will be sent if garbled data or illegal commands are received by PASHA. While PASHA will respond with a Negative Acknowledge mNAK '?' for certain communications errors, you can't be assured that it will always do so.

If there is a serial port error or communications timeout, or if there is a communications failure between PASHA and PADPU, you will get the Fail mFail 'X' message from PASHA as follows:

```
X.
```

If you send a valid message to PASHA, but PADPU is either not running, or a view file is not compiled and running, then PASHA will respond with a mNotReady message. For example if you sent:

```
SDeD48.
```

but the view file was not compiled, then you would receive back:

```
R.
```

C-like message definition

This section defines the message protocol using *C-like* declarations of constants and structures. The convention here is that words beginning with *c*, such as *cSETVALUE* and *cGETVALUE*, are character constants. Words beginning with *f*, such as *fUid* and *fVal*, are message field structures. Words beginning with *m*, such as *mSetValue* and *mGetValue*, are message structures.

```
//---- Message constants
const char cSETVALUE = 'S'; // an fType value
const char cGETVALUE = 'G'; // an fType value
const char cVALUE = 'V'; // an fType value
const char cNOTREADY = 'R'; // an fType value
const char cUNLISTEDUID = 'U'; // an fType value
const char cUNLISTEDCID = 'C'; // an fType value
const char cFAIL = 'X'; // an fType value
const char cNAK = '?'; // an fType value
const char cEOM = '.'; // an fEom value

//---- Message fields
/* fType: Message Type. 1 character, denotes the message type. */
struct fType {
  char data;
};

/* fUid: User ID. 3 ASCII characters, specifies one of the User IDs
entered as a Control Group name in the View File. */
struct fUid {
  char data[3];
};

/* fVal: Control Value. 2 ASCII hexadecimal digits, specifies a
control value between 0 and 255. */
struct fVal {
  char data[2];
};

/* fEom: End Of Message. 1 character, appears at the end of every
message, excepting mNak. */
```

```
struct fEom {
  char data = cEOM;
};

//---- Messages
/* mNak: Negative Acknowledge. Response sent to client upon receipt
of unintelligible data. This could be due to a communications error
or to data out of order. An mNak is not necessarily sent for every
byte of bad data. */
struct mNak {
  fType = cNAK;
};

/* mSetValue: Set Control Value. Request sent by a client to set the
value of a Control Group. */
struct mSetValue {
  fType type = cSETVALUE;
  fUid id;
  fVal val;
  fEom eom;
};

/* mGetValue: Get Control Value. Request sent by a client requesting
the value of a Control Group. */
struct mGetValue {
  fType type = cGETVALUE;
  fUid id;
  fEom eom;
};

/* mValue: Control Value. Response sent to the client acknowledging
an mGetValue or mSetValue. Note that it is possible, and normal in
some cases, that the val field will not match the val that was sent
in the mSetValue. */
struct mValue {
  fType type = cVALUE;
  fUid id;
  fVal val;
  fEom eom;
};

/* mUnlistedUid: Unlisted User ID error. Response indicating the
fUid specified in the mSetValue or mGetValue does not match any
Control Group name in the currently running View File. */
struct mUnlistedUid {
  fType type = cUNLISTEDUID;
  fUid id;
  fEom eom;
};
/* mUnlistedCID: Unlisted Control ID error. Response indicating a
control within the Control Group referenced by the fUid specified in
the mSetValue or mGetValue was not found in the currently compiled
View File. Probably means that the Control is "dead" or discarded.
*/
struct mUnlistedCID {
  fType type = cUNLISTEDCID;
  fUid id;
  fEom eom;
};
```

```
/* mNotReady: Not Ready. This means that there is no View File
currently compiled and running in MediaMatrix. */
struct mNotReady {
  fType type = cNOTREADY;
  fEom eom;
};

/* mFail: Something Has Failed. This is sent in response to serial
port errors, communication time-outs, and other internal errors not
covered directly. */
struct mFail {
  fType type = cFAIL;
  fEom eom;
};
```

Message structures quick chart

Message Name	**Message Fields**				
	Type	**Alphanumeric UID**	**Hex value**	**End of message**	**Style**
Set value	S	XXX	XX	.	Request
Get value	G	XXX		.	Request
Value	V	XXX	XX	.	Response
Not ready	R			.	Response
Unlisted UID	U	XXX		.	Response
Unlisted CID	C	XXX		.	Response
Fail	X			.	Response
Nak	?				Response

Real-time Concerns and Flow Control

In general, the serial control throughput into the MediaMatrix computer is dependent upon the number of Windows applications running, and the activity level of those applications. Any application which is actively animating a display on the screen will take time away from other applications, such as PASHA. MediaMatrix itself, if displaying any controls, requires a certain percentage of the processor power to keep those controls updated.

Serial data overruns at the MediaMatrix end can be prevented by having the external serial device wait for the response to each command it sends. If this is not possible, keep in mind that each PASHA service instance can store about 100 unacknowledged commands before overflowing its receive buffer.

Serial data overruns at the external serial device are usually not an issue since PASHA will never speak unless spoken to - the only way to elicit data from PASHA is to send a command.

A Note About Reading Meter Values

It is possible to read MediaMatrix audio meter values with PASHA, but there is an important fact to be aware of when doing so - as an optimization to save CPU cycles, meters are only updated when their controls are being displayed in the user interface. This implies that you must either have meter child windows open on the screen, or one control from each of the meters of interest must be copied out to an open child window or the main View Window. It is preferable to copy out a non-changing control, such as the Meter Time Constant, rather than the meter control itself, to save the user interface the burden of continuously updating the display. If having these controls visible to the user is a problem, they may be rendered invisible by manipulating their Control Object Properties (turn off the following check boxes: Text, Block, Bitmap and Style). Once you make a Control Object invisible it may be very difficult to find again unless you also check Fix Size and give it some nominal size, such as 6 by 6.

A numeric display of the peak hold value is not provided in the meter devices even though the algorithm calculates it. To read a meter's peak hold indicator value, start by copying the numeric control object (the yellow box with the dB value in it). Paste the copy down someplace convenient, perhaps in the meter device. While the new copy is selected, press Alt-Enter to bring up the Control Object Properties window, and select the Id tab. Under Control Id | Control, you will notice the digit entered is 2. If you change this to a 5 and click OK, the new numeric display will now read the peak hold indicator's value. It can then have a Control Group assigned in the usual way.

Serial Control Value to Device Control Value Tables

Some selected mappings of serial control values to device control values follow. You can also generate a table for any other control using a command in the MediaMatrix Terminal window in the following manner - after compiling a View File, twiddle the control of interest, switch to the Terminal window and press the 'v' on your keyboard. This will invoke the (V)alueTable command, which will print the table to the screen. You can also capture the table to a text file or printer by invoking the appropriate Terminal window modes.

Control value to dB's of gain table: -100dB to +12.0dB type control

0x00: -100dB	0x01: -99.3dB	0x02: -98.6dB	0x03: -97.8dB
0x04: -97.1dB	0x05: -96.4dB	0x06: -95.7dB	0x07: -95.0dB
0x08: -94.2dB	0x09: -93.5dB	0x0a: -92.8dB	0x0b: -92.1dB
0x0c: -91.3dB	0x0d: -90.6dB	0x0e: -89.9dB	0x0f: -89.2dB
0x10: -88.5dB	0x11: -87.7dB	0x12: -87.0dB	0x13: -86.3dB
0x14: -85.6dB	0x15: -84.8dB	0x16: -84.1dB	0x17: -83.4dB

0x18: -82.7dB	0x19: -82.0dB	0x1a: -81.2dB	0x1b: -80.5dB
0x1c: -79.8dB	0x1d: -79.1dB	0x1e: -78.4dB	0x1f: -77.6dB
0x20: -76.9dB	0x21: -76.2dB	0x22: -75.5dB	0x23: -74.7dB
0x24: -74.0dB	0x25: -73.3dB	0x26: -72.6dB	0x27: -71.9dB
0x28: -71.1dB	0x29: -70.4dB	0x2a: -69.7dB	0x2b: -69.0dB
0x2c: -68.2dB	0x2d: -67.5dB	0x2e: -66.8dB	0x2f: -66.1dB
0x30: -65.4dB	0x31: -64.6dB	0x32: -63.9dB	0x33: -63.2dB
0x34: -62.5dB	0x35: -61.8dB	0x36: -61.0dB	0x37: -60.3dB
0x38: -59.6dB	0x39: -58.9dB	0x3a: -58.1dB	0x3b: -57.4dB
0x3c: -56.7dB	0x3d: -56.0dB	0x3e: -55.3dB	0x3f: -54.5dB
0x40: -53.8dB	0x41: -53.1dB	0x42: -52.4dB	0x43: -51.7dB
0x44: -50.9dB	0x45: -50.2dB	0x46: -49.5dB	0x47: -48.8dB
0x48: -48.0dB	0x49: -47.3dB	0x4a: -46.6dB	0x4b: -45.9dB
0x4c: -45.2dB	0x4d: -44.4dB	0x4e: -43.7dB	0x4f: -43.0dB
0x50: -42.3dB	0x51: -41.6dB	0x52: -40.8dB	0x53: -40.1dB
0x54: -39.4dB	0x55: -38.7dB	0x56: -37.9dB	0x57: -37.2dB
0x58: -36.5dB	0x59: -35.8dB	0x5a: -35.1dB	0x5b: -34.3dB
0x5c: -33.6dB	0x5d: -32.9dB	0x5e: -32.2dB	0x5f: -31.5dB
0x60: -30.7dB	0x61: -30.0dB	0x62: -29.3dB	0x63: -28.6dB
0x64: -27.8dB	0x65: -27.1dB	0x66: -26.4dB	0x67: -25.7dB
0x68: -25.0dB	0x69: -24.2dB	0x6a: -23.5dB	0x6b: -22.8dB
0x6c: -22.1dB	0x6d: -21.3dB	0x6e: -20.6dB	0x6f: -19.9dB
0x70: -19.2dB	0x71: -18.5dB	0x72: -17.7dB	0x73: -17.0dB
0x74: -16.3dB	0x75: -15.6dB	0x76: -14.9dB	0x77: -14.1dB
0x78: -13.4dB	0x79: -12.7dB	0x7a: -12.0dB	0x7b: -11.2dB
0x7c: -10.5dB	0x7d: -9.80dB	0x7e: -9.08dB	0x7f: -8.36dB

0x80: -7.92dB	0x81: -7.77dB	0x82: -7.61dB	0x83: -7.45dB
0x84: -7.29dB	0x85: -7.14dB	0x86: -6.98dB	0x87: -6.82dB
0x88: -6.67dB	0x89: -6.51dB	0x8a: -6.35dB	0x8b: -6.20dB
0x8c: -6.04dB	0x8d: -5.88dB	0x8e: -5.73dB	0x8f: -5.57dB
0x90: -5.41dB	0x91: -5.26dB	0x92: -5.10dB	0x93: -4.94dB
0x94: -4.78dB	0x95: -4.63dB	0x96: -4.47dB	0x97: -4.31dB
0x98: -4.16dB	0x99: -4.00dB	0x9a: -3.84dB	0x9b: -3.69dB
0x9c: -3.53dB	0x9d: -3.37dB	0x9e: -3.22dB	0x9f: -3.06dB
0xa0: -2.90dB	0xa1: -2.75dB	0xa2: -2.59dB	0xa3: -2.43dB
0xa4: -2.27dB	0xa5: -2.12dB	0xa6: -1.96dB	0xa7: -1.80dB
0xa8: -1.65dB	0xa9: -1.49dB	0xaa: -1.33dB	0xab: -1.18dB
0xac: -1.02dB	0xad: -0.86dB	0xae: -0.70dB	0xaf: -0.55dB
0xb0: -0.39dB	0xb1: -0.24dB	0xb2: -0.08dB	0xb3: +0.08dB
0xb4: +0.24dB	0xb5: +0.39dB	0xb6: +0.55dB	0xb7: +0.71dB
0xb8: +0.86dB	0xb9: +1.02dB	0xba: +1.18dB	0xbb: +1.33dB
0xbc: +1.49dB	0xbd: +1.65dB	0xbe: +1.80dB	0xbf: +1.96dB
0xc0: +2.12dB	0xc1: +2.27dB	0xc2: +2.43dB	0xc3: +2.59dB
0xc4: +2.75dB	0xc5: +2.90dB	0xc6: +3.06dB	0xc7: +3.22dB
0xc8: +3.37dB	0xc9: +3.53dB	0xca: +3.69dB	0xcb: +3.84dB
0xcc: +4.00dB	0xcd: +4.16dB	0xce: +4.31dB	0xcf: +4.47dB
0xd0: +4.63dB	0xd1: +4.78dB	0xd2: +4.94dB	0xd3: +5.10dB
0xd4: +5.26dB	0xd5: +5.41dB	0xd6: +5.57dB	0xd7: +5.73dB
0xd8: +5.88dB	0xd9: +6.04dB	0xda: +6.20dB	0xdb: +6.35dB
0xdc: +6.51dB	0xdd: +6.67dB	0xde: +6.82dB	0xdf: +6.98dB
0xe0: +7.14dB	0xe1: +7.29dB	0xe2: +7.45dB	0xe3: +7.61dB

0xe4: +7.77dB	0xe5: +7.92dB	0xe6: +8.08dB	0xe7: +8.24dB
0xe8: +8.39dB	0xe9: +8.55dB	0xea: +8.71dB	0xeb: +8.86dB
0xec: +9.02dB	0xed: +9.18dB	0xee: +9.33dB	0xef: +9.49dB
0xf0: +9.65dB	0xf1: +9.80dB	0xf2: +9.96dB	0xf3: +10.1dB
0xf4: +10.3dB	0xf5: +10.4dB	0xf6: +10.6dB	0xf7: +10.7dB
0xf8: +10.9dB	0xf9: +11.1dB	0xfa: +11.2dB	0xfb: +11.4dB
0xfc: +11.5dB	0xfd: +11.7dB	0xfe: +11.8dB	0xff: +12.0dB

Control value to dB's of gain table: -100dB to +0.00dB type control

0x00: -100dB	0x01: -99.4dB	0x02: -98.7dB	0x03: -98.1dB
0x04: -97.5dB	0x05: -96.9dB	0x06: -96.2dB	0x07: -95.6dB
0x08: -95.0dB	0x09: -94.4dB	0x0a: -93.7dB	0x0b: -93.1dB
0x0c: -92.5dB	0x0d: -91.8dB	0x0e: -91.2dB	0x0f: -90.6dB
0x10: -90.0dB	0x11: -89.3dB	0x12: -88.7dB	0x13: -88.1dB
0x14: -87.5dB	0x15: -86.8dB	0x16: -86.2dB	0x17: -85.6dB
0x18: -84.9dB	0x19: -84.3dB	0x1a: -83.7dB	0x1b: -83.1dB
0x1c: -82.4dB	0x1d: -81.8dB	0x1e: -81.2dB	0x1f: -80.6dB
0x20: -79.9dB	0x21: -79.3dB	0x22: -78.7dB	0x23: -78.0dB
0x24: -77.4dB	0x25: -76.8dB	0x26: -76.2dB	0x27: -75.5dB
0x28: -74.9dB	0x29: -74.3dB	0x2a: -73.6dB	0x2b: -73.0dB
0x2c: -72.4dB	0x2d: -71.8dB	0x2e: -71.1dB	0x2f: -70.5dB
0x30: -69.9dB	0x31: -69.3dB	0x32: -68.6dB	0x33: -68.0dB
0x34: -67.4dB	0x35: -66.7dB	0x36: -66.1dB	0x37: -65.5dB

0x38: -64.9dB	0x39: -64.2dB	0x3a: -63.6dB	0x3b: -63.0dB
0x3c: -62.4dB	0x3d: -61.7dB	0x3e: -61.1dB	0x3f: -60.5dB
0x40: -59.8dB	0x41: -59.2dB	0x42: -58.6dB	0x43: -58.0dB
0x44: -57.3dB	0x45: -56.7dB	0x46: -56.1dB	0x47: -55.5dB
0x48: -54.8dB	0x49: -54.2dB	0x4a: -53.6dB	0x4b: -52.9dB
0x4c: -52.3dB	0x4d: -51.7dB	0x4e: -51.1dB	0x4f: -50.4dB
0x50: -49.8dB	0x51: -49.2dB	0x52: -48.5dB	0x53: -47.9dB
0x54: -47.3dB	0x55: -46.7dB	0x56: -46.0dB	0x57: -45.4dB
0x58: -44.8dB	0x59: -44.2dB	0x5a: -43.5dB	0x5b: -42.9dB
0x5c: -42.3dB	0x5d: -41.6dB	0x5e: -41.0dB	0x5f: -40.4dB
0x60: -39.8dB	0x61: -39.1dB	0x62: -38.5dB	0x63: -37.9dB
0x64: -37.3dB	0x65: -36.6dB	0x66: -36.0dB	0x67: -35.4dB
0x68: -34.7dB	0x69: -34.1dB	0x6a: -33.5dB	0x6b: -32.9dB
0x6c: -32.2dB	0x6d: -31.6dB	0x6e: -31.0dB	0x6f: -30.4dB
0x70: -29.7dB	0x71: -29.1dB	0x72: -28.5dB	0x73: -27.8dB
0x74: -27.2dB	0x75: -26.6dB	0x76: -26.0dB	0x77: -25.3dB
0x78: -24.7dB	0x79: -24.1dB	0x7a: -23.4dB	0x7b: -22.8dB
0x7c: -22.2dB	0x7d: -21.6dB	0x7e: -20.9dB	0x7f: -20.3dB
0x80: -19.9dB	0x81: -19.8dB	0x82: -19.6dB	0x83: -19.5dB
0x84: -19.3dB	0x85: -19.1dB	0x86: -19.0dB	0x87: -18.8dB
0x88: -18.7dB	0x89: -18.5dB	0x8a: -18.4dB	0x8b: -18.2dB
0x8c:-18.0dB	0x8d: -17.9dB	0x8e: -17.7dB	0x8f: -17.6dB

0x90: -17.4dB	0x91: -17.3dB	0x92: -17.1dB	0x93: -16.9dB
0x98: -16.2dB	0x99: -16.0dB	0x9a: -15.8dB	0x9b: -15.7dB
0x9c: -15.5dB	0x9d: -15.4dB	0x9e: -15.2dB	0x9f: -15.1dB
0xa0: -14.9dB	0xa1: -14.7dB	0xa2: -14.6dB	0xa3: -14.4dB
0xa4: -14.3dB	0xa5: -14.1dB	0xa6: -14.0dB	0xa7: -13.8dB
0xa8: -13.6dB	0xa9: -13.5dB	0xaa: -13.3dB	0xab: -13.2dB
0xac: -13.0dB	0xad: -12.9dB	0xae: -12.7dB	0xaf: -12.5dB
0xb0: -12.4dB	0xb1: -12.2dB	0xb2: -12.1dB	0xb3: -11.9dB
0xb4: -11.8dB	0xb5: -11.6dB	0xb6: -11.5dB	0xb7: -11.3dB
0xb8: -11.1dB	0xb9: -11.0dB	0xba: -10.8dB	0xbb: -10.7dB
0xbc: -10.5dB	0xbd: -10.4dB	0xbe: -10.2dB	0xbf: -10.0dB
0xc0: -9.88dB	0xc1: -9.73dB	0xc2: -9.57dB	0xc3: -9.41dB
0xc4: -9.25dB	0xc5: -9.10dB	0xc6: -8.94dB	0xc7: -8.78dB
0xc8: -8.63dB	0xc9: -8.47dB	0xca: -8.31dB	0xcb: -8.16dB
0xcc: -8.00dB	0xcd: -7.84dB	0xce: -7.69dB	0xcf: -7.53dB
0xd0: -7.37dB	0xd1: -7.22dB	0xd2: -7.06dB	0xd3: -6.90dB
0xd4: -6.74dB	0xd5: -6.59dB	0xd6: -6.43dB	0xd7: -6.27dB
0xd8: -6.12dB	0xd9: -5.96dB	0xda: -5.80dB	0xdb: -5.65dB
0xdc: -5.49dB	0xdd: -5.33dB	0xde: -5.18dB	0xdf: -5.02dB
0xe0: -4.86dB	0xe1: -4.71dB	0xe2: -4.55dB	0xe3: -4.39dB
0xe4: -4.23dB	0xe5: -4.08dB	0xe6: -3.92dB	0xe7: -3.76dB
0xe8: -3.61dB	0xe9: -3.45dB	0xea: -3.29dB	0xeb: -3.14dB

0xec: -2.98dB	0xed: -2.82dB	0xee: -2.67dB	0xef: -2.51dB
0xf0: -2.35dB	0xf1: -2.20dB	0xf2: -2.04dB	0xf3: -1.88dB
0xf4: -1.72dB	0xf5: -1.57dB	0xf6: -1.41dB	0xf7: -1.25dB
0xf8: -1.10dB	0xf9: -0.94dB	0xfa: -0.78dB	0xfb: -0.63dB
0xfc: -0.47dB	0xfd: -0.31dB	0xfe: -0.16dB	0xff: +0.00dB

Control value to Router input selection table: 1x1 Router

Input	**Value Range**
Off	0x00 - 0x7F
1	0x80 - 0xFF

Control value to Router input selection table: 2x1 Router

Input	**Value Range**
Off	0x00 - 0x3F
1	0x40 - 0xBF
2	0xC0 - 0xFF

Control value to Router input selection table: 4x1 Router

Input	**Value Range**
Off	0x00 - 0x1F
1	0x20 - 0x5F
2	0x60 - 0x9F
3	0xA0 - 0xDF
4	0xE0 - 0xFF

Control value to Router input selection table: 8x1 Router

Input	**Value Range**
Off	0x00 - 0x0F
1	0x10 - 0x2F
2	0x30 - 0x4F
3	0x50 - 0x6F
4	0x70 - 0x8F
5	0x90 - 0xAF
6	0xB0 - 0xCF
7	0xD0 - 0xEF
8	0xF0 - 0xFF

Chapter 4

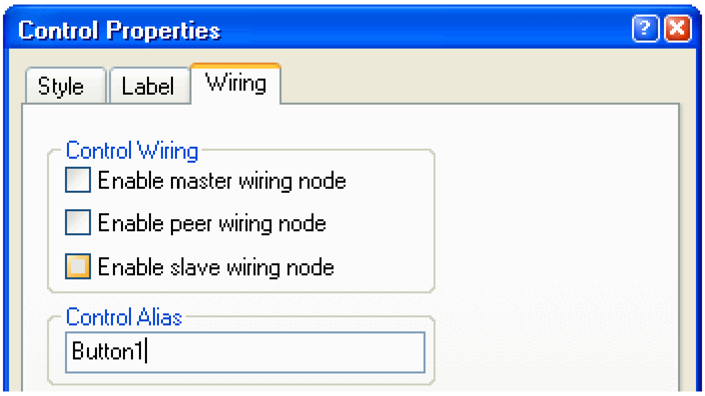
RATC

In This Chapter

Introduction

RATC, or Remote Access Terminal Control, is a command-line based protocol that allows a remote client program to set and get the control values in a NION project. The remote client communicates with a project via a TCP network connection to any one of the NioNodes in the project, so RATC is compatible with both local area networks and the Internet. Any number of RATC clients (within reason) may connect simultaneously to any NioNode in the project. The RATC1 or RATC2 service is configured and enabled through the NioNode device properties.

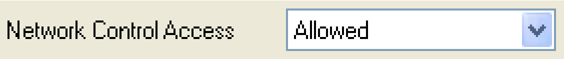
You assign controls to be available to RATC1, RATC2 and/or PASHA with Control Aliases. PASHA only supports 3 character and 4 numeral aliases, but RATC1 and RATC2 support aliases of several dozen characters, including spaces. Remember that only one control in a project can have a particular Control Alias. If you assign another control the same alias, the other control with that alias loses the alias. If you need multiple controls to have control from the same alias, you should assign the alias to one of the controls and use control wiring to connect them within the project.



☐ Enable master wiring node
☐ Enable peer wiring node
☐ Enable slave wiring node

Control Alias: Button1

You enable RATC communications project-wide in the User Accounts properties area in the File menu as seen below. For more information on this, see the User Accounts topic. This is different from the Telnet option in the config menu on the NION front panel. This property extends to all NioNodes that are members of the project.


Network Control Access: Allowed

Differences Between RATC in Classic MediaMatrix and NION

Classic MediaMatrix supported RATC v0.9 on TCP/IP connections. NION supports this newer and more flexible protocol, RATC v2, which can run on either a TCP/IP connection or a serial connection. We refer to them as RATC1 and RATC2.

For backward compatibility, NION supports RATC v1, which is compatible with MediaMatrix's RATC v0.9. NION also features a version of PASHA that is compatible with the service of the same name on Classic MediaMatrix.

It is recommended that RATC v2, over TCP/IP, be used whenever possible.

One difference between NION external control and MediaMatrix external control is that in NION, the external control services only run while the Project is running. This is a little different than in Classic MediaMatrix, where the control services run whenever the MWare application is running, independent of whether a View file is compiled and running or not.

Using TELNET with RATC

TELNET is a computer industry standard network communications protocol which is generally used to connect remote users to a host computer using a text-based interface. The user can then control certain functions on that host computer. TELNET works over both local area networks and the Internet.

RATC1 can function as a TELNET server. This means that any standard TELNET client program (such as that which ships with Windows NT) may be used to manipulate and monitor Control Group values in MediaMatrix. This is most useful for verifying that things are configured such that a successful RATC connection can be made to the MediaMatrix computer, since it is assumed that a command-line style interface will be enjoyed only by few creaky old UNIX gurus.

To connect a TELNET terminal to RATC1, one simply specifies to the TELNET client program the MediaMatrix computer network name or IP address and the TCP port number that RATC has been assigned in the Remote Services dialog box. The Internet Assigned Numbers Authority port assignment for RATC is 1632, keyword pammratc.

RATC does not echo data input from the client, so it is advised that local echo be enabled on the TELNET client program so that one can see what one is typing while one types it so as to reduce ones typographical error frequency.

Note: The BEL character that precedes each error response from RATC may very well cause a bell sound in your TELNET client computer.

RATC supports only the minimum TELNET protocol, and thus will refuse any and all TELNET option requests coming from a TELNET client program.

Commands and Responses

Each RATC2 command is an ASCII text string terminated with an ASCII CR. Each command results in a response from RATC1. Except during login, RATC1 will send no unsolicited data to a client. Each response from RATC1 is an ASCII text string terminated with both an ASCII CR and NL character. Each response that indicates an error in the input command line is preceded by an ASCII BEL character.

RATC1 commands are not case sensitive, but the password string is case sensitive (as per the MediaMatrix security model). RATC1 commands are, however, presented here with mixed case to improve readability.

Control codes other than CR in the command line are ignored, except for the following two exceptions: ASCII BS (backspace) is supported to make using TELNET more reasonable; and the TELNET option control escape sequence protocol is automatically dealt with (all TELNET option requests are refused by the RATC1 service).

Name	ASCII Name	C++ Name	Decimal Value	Hexadecimal Value
carriage return	CR	\r	13	D
newline	NL or LF	\n	10	A

Name	ASCII Name	C++ Name	Decimal Value	Hexadecimal Value
alert	BEL	\a	7	7
backspace	BS	\b	8	8

RATC1 commands and responses are designed to be compatible with two use scenarios: computer control with a software application and human control with TELNET. To aid in the text parsing required for computer control, the non-error responses all have a unique first character, and the error responses all have an ASCII BEL character followed by a unique character.

Change Groups

If many Control Alias values are to be monitored by the client software, efficiency is gained by using the Change Group commands. These allow a set of Control Aliases to be queried for value changes with a single command. When a Control Alias is added to the Change Group, it is considered initially 'changed', and a subsequent get command will cause its most current value to be returned. RATC2 also provides the ability to have multiple Change Groups.

At the initial connect, and after each redeploy or re-emulate, there will be no Change Groups because they are cleared between deployments. If there are no Change groups assigned when a Change Group Get command is received, a default change group is created with no controls, so the response would show zero changes. It is not necessary that the client software clear the Change Group before disconnecting - this is done automatically by the NION.

RATC1 commands

help Command

Usage

As an aid in the TELNET operation of RATC1, the following command will result in a "helpful" display listing the RATC1 command set

```
help\r
```

Response

The help response starts with

```
{
```

and the final line of the response is:

```
}\r\n
```

as an aid in allowing a computer program to ignore the contents of the help response.

Note: Parsing of the help and list command responses by a software-based client is strongly discouraged since the formats are subject to change.

Possible Error Messages

```
\aOverflow\r\n
```

statusGet command

Usage

Gets the state and name of the project running on this NioNode project member.
The client issues:

```
statusGet\r
```

Response

RATC1 responds with:

```
statusIs running "Level 1 Ballrooms"\r\n
```

If no project is running, you will not be able to connect, so there is no alternative response here (no "stopped or not running" status).

Note: The file name is enclosed in quotes to support spaces in the file name. In future versions, we plan to add other states for the second part of the response for other situations.

Possible Error Messages

```
\aOverflow\r\n
```

controlGet command

Usage

The *controlGet* command is used to determine the value of a Control Alias in a running project. For example:

```
controlGet "Main Gain"\r
```

Response

On success, RATC1 responds with:

```
valueIs "Main Gain" +0.00dB 70.0%\r\n
```

Possible Error Messages

```
\aBadArgumentCount\r\n
```

```
\aOverflow\r\n
```

```
\aUnlistedGroup "AliasName"\r\n
```

In the context of NION, *UnlistedGroup* refers to the lack of a control with that control alias. In this instance, the term group is used for compatibility purposes with Classic MediaMatrix RATC.

controlSet command

Usage

The *controlSet* command is used to set the value of an aliased control in a deployed project. For example:

```
controlSet "Main Gain" -3.4\r
```

Response

On success, RATC1 responds with:

```
valueIs "Main Gain" -3.40dB 61.5%\r\n
```

The Control Alias name is enclosed in quotes to support names with embedded spaces. In fact, the quotes are not required for the command argument if the Control alias has no embedded spaces.

In the response, the first token after the alias name is the current value expressed in the units appropriate to that particular Control Alias. In the response, the second token after the Control Alias name is the current value of the alias expressed as a percentage of the maximum value. This can be thought of as a physical knob position - in this example, the 61.5% knob position corresponds to -3.4dB, and a 100% knob position will correspond to +12dB.

Possible Error Messages

```
\aBadArgumentCount\r\n
```

```
\aOverflow\r\n
```

```
\aUnlistedGroup "GroupName"\r\n
```

In the context of NION, *UnlistedGroup* refers to the lack of a control with that control alias. In this instance, the term group is used for compatibility purposes with Classic MediaMatrix RATC.

controlList command

Usage

As an aid in the TELNET operation of RATC1, the following command:

```
controlList\r
```

will result in a display listing the current set of Control Alias names.

Response

The list response starts with:

```
{
```

and the final line of the response is:

```
}\r\n
```

as an aid in allowing a computer program to ignore the contents of the list response.

Note: There may be other matching brace characters within the list response.

Note: Parsing of the help and list command responses by a software-based client is strongly discouraged since the formats are subject to change.

Possible error messages

```
\aOverflow\r\n
```

changeGroupAddControl command

Usage

A Control Alias is added to the Change Group with the `changeGroupAddControl` command. For example:

```
changeGroupAddControl "Main Gain"\r
```

Response

On success, RATC1 responds with:

```
addedToChangeGroup "Main Gain"\r\n
```

Possible error messages

```
\aBadArgumentCount\r\n
```

```
\aOverflow\r\n
```

```
\aInvalidChangeGroup
```

```
\aUnlistedGroup "GroupName"\r\n
```

In the context of NION, 'UnlistedGroup' refers to the lack of a control with that control alias. In this instance, the term group is used for compatibility purposes with Classic MediaMatrix RATC.

changeGroupRemoveControl command

Usage

A Control Alias is removed from the Change Group with the *changeGroupRemoveControl* command. For example:

```
changeGroupRemoveControl "Main Gain"\r
```

Response

On success, RATC1 responds with:

```
removedFromChangeGroup "Main Gain"\r\n
```

The above response is issued even if the Control Alias named was not in the Change Group.

Possible error messages

```
\aBadArgumentCount\r\n
```

```
\aOverflow\r\n
```

```
\aInvalidChangeGroup
```

```
\aUnlistedGroup "GroupName"\r\n
```

In the context of NION, 'UnlistedGroup' refers to the lack of a control with that control alias. In this instance, the term group is used for compatibility purposes with Classic MediaMatrix RATC.

changeGroupRemoveControl command

Usage

The Change Group is cleared of all Control Aliases with the command:

```
changeGroupClear\r
```

Response

On success, RATC1 responds with:

```
clearedChangeGroup\r\n
```

Possible error messages

```
\aOverflow\r\n
```

changeGroupGet command

Usage

The Control Groups in the Change Group are polled for value changes with the command:

```
changeGroupGet n\r
```

where n is the number of changes the client wishes to read. If n is not supplied, all values that have changed will be returned. If there are more changed values than are requested, subsequent *changeGroupGet* commands will return them.

Response

On success, RATC1 responds with the minimum of the number of values that have changed and number of changes that were requested, and then a value for each change. For example:

```
numberOfChanges 2\r\n

valueIs "Main Gain" -15.2dB 46.1%\r\n

valueIs "Channel1Gain" -20.0dB 50.0%\r\n
```

In this example, there are two Control Aliases in the Change Group that have changed since the last query. If the command is issued again, with no changes to Control Aliases, or if there are no Control Aliases in the Change Group, the response will be:

```
numberOfChanges 0\r\n
```

Possible error messages

```
\aBadArgumentCount\r\n

\aOverflow\r\n

\aNotRunning\r\n

\aInvalidChangeGroup
```

RATC1 responses

`statusIs`

`valueIs`

`addedToChangeGroup`

`removedFromChangeGroup`

`clearedChangeGroup`

`numberOfChanges`

There are also various error responses.

In addition, the RATC1 login process uses a name prompt, a password prompt, a version statement and a welcome statement.

Chapter 5

RATC2

In This Chapter

Introduction

RATCv2 is Telnet compatible, meaning that it is text-based and that is possible to use a Telnet client program to drive it. Each command, with arguments, should be issued followed by a CR and/or LF. Each response is followed by CR/LF. Each command and response is a single token with no spaces, and each command argument or response value is either a single token or is enclosed in double-quotes.

Each command has an abbreviated shortcut formed from the first letter of each word within the command.

Command list

command	**short version**	**purpose**
help	h	display the list of commands
logIn name password	li	log in with username and password
statusGet	sg	report status
keepAlive seconds	ka	disconnect if no activity in n seconds
quietModeEnable	qme	suppress responses from non-query commands
quietModeDisable	qmd	do not suppress responses
controlGet control	cg	get a Control's value and position
controlSet control value	cs	set a Control's value
controlPositionSet control position	cps	set a Control's position (0.00-1.00)
changeGroupControlAdd [group] control	cgca	add a Control to a Change Group
changeGroupControlRemove [group] control	cgcr	remove a Control from a Change Group
changeGroupGet [group]	cgg	get changed values from a Change Group
changeGroupClear [group]	cgc	clear a Change Group (of changed values)

command	short version	purpose
changeGroupSchedule [group] seconds	cgs	schedule recurring Change Group gets

RATC2 responses

```
statusIs
valueIs
loggedIn
keepAlive
quietModeEnabled
quietModeDisabled
changeGroupControlAdded
changeGroupControlRemoved
changeGroupCleared
changeGroupChanges
changeGroupSchedule
```

RATC2 error responses

The complete list of error responses is as follows:

```
badCommand
badArgumentCount
overflow
unlistedControl
invalidChangeGroup
commandFailed
commandUnsupported
notLoggedIn
loginFailed
```

Note: The RATC2 login process does not display a username and password prompt. To login, use the *li* command.

Commands in detail

help Command

Command	help
Shortcut	h
Arguments	none
Availability	always
Purpose	displays a list of commands
Notes	The number of line/commands to expect is given in the first line
Response	a list of the available commands

Usage example

```
help\r
```

or

```
h\r
```

Response

```
RATCv2.0 Help 15
h    help : display this help list
li   logIn name password : log in with a password
sg   statusGet : report status
ka   keepAlive seconds : disconnect if no activity in n seconds
qme  quietModeEnable  : suppress responses from non-query commands
qmd  quietModeDisable : allow responses from all commands
cl   controlList : get the list of available Controls
cg   controlGet control : get a Controls value
cs   controlSet control value : set a Controls value
cps  controlPositionSet control value : set a Controls position
(0.00-1.00)
cgca changeGroupControlAdd [group] control : add a Control to a
Change Group
cgcr changeGroupControlRemove [group] control : remove a Control
from a Change Group
cgg  changeGroupGet [group] : get changed values from a Change Group
cgc  changeGroupClear [group] : clear a Change Group (of changed
values)
cgs  changeGroupSchedule [group] seconds : schedule recurring Change
Group gets
```

Note: Parsing of the help and list command responses by a software-based client is strongly discouraged since the formats are subject to change.

Possible Error Messages

```
\aOverflow\r\n
```

login command

Command	logIn
Shortcut	li
Arguments	<username> <password> If password is blank, it can be omitted. If both username and password are blank, both can be omitted.
Availability	always
Purpose	security
Notes	you are not prompted to log on, but must instead explicitly issue the login command. A few commands are available prior to logging in: help, quietModeEnable, quietModeDisable. If any other command is attempted prior to logging in, the response will be notLoggedIn.
Response	loggedIn or loginFailed

Usage example

If the client's username is *maintenance* and the password is *youguessedit*, the client should type:

```
logIn maintenance youguessedit\r
```

or

```
li maintenance youguessedit\r
```

Response

```
loggedIn
```

Possible error messages

```
\aloginFailed\r\n
```

```
\aOverflow\r\n
```

statusGet command

Command	statusGet
Shortcut	sg
Arguments	none
Availability	always
Purpose	get the current state of the system
Response	Something like this: statusIs running "Your Project Here"

Usage Example

The client issues:

```
statusGet\r
```

or

```
sg\r
```

Response

```
statusIs running "Level 1 Ballrooms"\r\n
```

If no project is running, you will not be able to connect, so there is no alternative response here (no "stopped or not running" status).

Note: The file name is enclosed in quotes to support spaces in the file name. In future versions, we plan to add other states for the second part of the response for other situations.

Possible Error Messages

```
\aOverflow\r\n
```

keepAlive command

Command	keepAlive
Shortcut	ka
Arguments	seconds
Availability	when logged in
Purpose	Starts a watchdog timer, requiring the client to communicate within that period or be disconnected. It is recommended that keepAlive be used to ensure that TCP/IP client connections get closed in the face of network failures (or even just unplugging and plugging in network connections). If the network goes down and the client attempts to communicate, the client will get a timeout and disconnect, reconnecting once the network is restored. The server, however, would normally have no way of knowing that the client gave up on its first connection and reconnected, so server network stack resources consumed over time. keepAlive solves this problem.
Notes	You can disable the watchdog timer with an argument of zero. This command applies to TCP/IP only, not to the serial port.
Response	Something like this: keepAlive 10

Usage Example

The client issues:

```
keepAlive 120\r
```

or

```
ka 120\r
```

Response

```
keepAlive 120\r\n
```

Possible Error Messages

```
\aOverflow\r\n
```

```
\aBadArgumentCount\r\n
```

```
\anotLoggedIn\r\n
```

quietModeEnable command

Command	quietModeEnable
Shortcut	qme
Arguments	none
Availability	always
Purpose	Suppresses the server responses to some of the commands: logIn, controlSet, changeGroupControlAdd, changeGroupControlRemove, changeGroupClear, keepAlive, and quietModeEnable.
Notes	
Response	none!

Usage Example

The client issues:

```
quietModeEnable\r
```

or

```
qme\r
```

Response

RATC2 does not respond.

Possible Error Messages

```
\aOverflow\r\n
```

quietModeDisable command

Command	quietModeDisable
Shortcut	qmd
Arguments	none
Availability	always
Purpose	To leave quiet mode, restoring responses to all commands
Response	quietModeDisabled

Usage Example

The client issues:

```
quietModeDisable\r
```

or

```
qmd\r
```

Response

```
quietModeDisabled\r\n
```

Possible Error Messages

```
\aOverflow\r\n
```

controlGet command

Command	controlGet
Shortcut	cg
Arguments	control alias
Availability	when logged in
Purpose	To read the current value and position of a control
Notes	This command returns the control alias name, the string value of the control (such as "6.2dB"), and the positional value (0.000 through 1.000). When the control name has spaces, the argument should be enclosed in double-quotes, as in: controlGet "Master Gain"
Response	Something like this: valueIs "Master Gain" -90.0dB 0.100

Usage Example

The client issues:

```
controlGet "Main Gain"\r
```

or

```
cg "Main Gain"\r
```

The Control Alias name is enclosed in quotes to support names with embedded spaces. In fact, the quotes are not required for the command argument if the Control name has no embedded spaces.

Response

```
valueIs "Main Gain" 1.99dB 0.864\r\n
```

Note: The positional scale of 0.864 is on a 0.000 to 1.000 scale, but can be read as 86.4% of the control's range. The first value after the alias name is the exact reading of the control (known as the "string value") and will differ, depending on the control type, however, all responses use this 0.000 to 1.000 scale for the last element of the response.

Possible Error Messages

```
\aBadArgumentCount\r\n
```

```
\anotLoggedIn\r\n
```

```
\aUnlistedControl "AliasName"\r\n
```

```
\aOverflow\r\n
```

controlSet command

Command	controlSet
Shortcut	cs
Arguments	control value
Availability	when logged in
Purpose	To set the string value of a control
Notes	Like controlGet, this command returns the Control name, the string value of the Control, and the positional value. When the Control name has spaces, the argument should be enclosed in double-quotes, as in: controlSet "Master Gain" -75
Response	Something like this: valueIs "Master Gain -75.0dB 0.250

Usage Example

The client issues:

```
controlSet "Main Gain" -3.4\r
```

or

```
cs "Main Gain" -3.4\r
```

The Control Alias name is enclosed in quotes to support names with embedded spaces. In fact, the quotes are not required for the command argument if the Control name has no embedded spaces.

Response

```
valueIs "Main Gain" -3.50dB 0.818\r\n
```

In the response, the first token after the alias name is the current value expressed in the units appropriate to that particular control. In the response, the second token after the Control Alias name is the current value of the alias expressed as a percentage of the maximum value expressed as a decimal value between 0 and 1. This can be thought of as a physical knob position - in this example, the 0.615 (61.5%) knob position corresponds to -3.4dB, and a 1.000 (100%) knob position will correspond to +12dB.

Possible Error Messages

```
\aBadArgumentCount\r\n
```

```
\anotLoggedIn\r\n
```

```
\aUnlistedControl "AliasName"\r\n
```

```
\aOverflow\r\n
```

controlPositionSet command

Command	controlPositionSet
Shortcut	cps
Arguments	control position.
Availability	when logged in
Purpose	to set the positional value of a control, corresponding to a "slider position" between 0 and 1. For example, .535 is a position of 53.5%.
Notes	Like controlGet and controlSet, this command returns the control name, the string value and the positional value of the control. When the control alias has spaces, the argument should be enclosed in double-quotes, as in: controlSet "Master Gain" .25
Response	Something like this: valueIs "Master Gain" -75.0dB 0.250

Usage Example

The client issues::

```
controlPositionSet "Master Gain" .25\r
```

or

```
cps "Master Gain" .25\r
```

Response

```
valueIs "Master Gain" -75.0dB 0.250\r\n
```

Possible Error Messages

```
\aBadArgumentCount\r\n
\anotLoggedIn\r\n
\aOverflow\r\n
```

controlList Command

Command	controlList
Shortcut	cl
Arguments	none
Availability	when logged in
Purpose	to list controls that can be accessed
Response	presents a list of all control aliases in double-quotes with CR/LF after each

Usage Example

The client issues:

```
controlList\r
```

or

```
cl\r
```

which results in a response listing the current set of Control Alias names.

Response

If the aliases are: "control1 ... control5", then the response would be:

```
"control1"\r\n
"control2"\r\n
"control3"\r\n
"control4"\r\n
"control5"\r\n
```

Note: Parsing of the help and list command responses by a software-based client is strongly discouraged since the formats are subject to change.

Possible Error Messages

```
\anotLoggedIn\r\n
```

```
\aOverflow\r\n
```

changeGroupControlAdd command

Command	changeGroupControlAdd
Shortcut	cgca
Arguments	control [group]
Availability	when logged in
Purpose	To add a control to a Change Group. If the Change Group named does not yet exist, it is created.
Notes	The group name argument is optional. If not included, a Change Group named 'default Change Group' will be used.
Response	changeGroupControlAdded

Usage example

The client issues:

```
changeGroupControlAdd "Main Gain"\r
```

or

```
cgca "Main Gain"\r
```

Response

```
changeGroupControlAdded\r\n
```

Possible Error Messages

```
\aBadArgumentCount\r\n
\anotLoggedIn\r\n
\aUnlistedControl "AliasName"\r\n
\aOverflow\r\n
```

changeGroupControlRemove Command

Command	changeGroupControlRemove
Shortcut	cgcr
Arguments	control [group].
Availability	when logged in
Purpose	to remove a control from a Change Group.
Notes	The group name argument is optional. If not included, a Change Group named 'default Change Group' will be used. If the Change Group named does not exist, the response is something like invalidChangeGroup "yourBogusGroupName"
Response	changeGroupControlAdded

Usage Example

A Control Alias is removed from the Change Group with the *changeGroupRemoveControl* command. For example:

```
changeGroupControlRemove "Main Gain"\r
```

or

```
cgcr "Main Gain"\r
```

Response

```
changeGroupControlRemoved "Main Gain"\r\n
```

The above response is issued even if the Control Alias named was not in the Change Group.

Possible Error Messages

```
\aBadArgumentCount\r\n
```

```
\anotLoggedIn\r\n
```

```
\aInvalidChangeGroup "yourBogusGroupName"\r\n
```

```
\aUnlistedControl "AliasName"\r\n
```

```
\aOverflow\r\n
```

changeGroupClear Command

Command	changeGroupClear
Shortcut	cgc
Arguments	[group].
Availability	when logged in
Purpose	to destroy a Change Group.
Notes	The group name argument is optional. If not included, a Change Group named 'default Change Group' will be used. It is not necessary to remove the controls from a Change Group before destroying it.
Response	changeGroupCleared

Usage example

The Change Group is cleared of all Control Aliases with the command:

```
changeGroupClear\r
```

or

```
cgc\r
```

Response

```
changeGroupCleared\r\n
```

Possible Error Messages

```
\aBadArgumentCount\r\n
\anotLoggedIn\r\n
\aInvalidChangeGroup "yourBogusGroupName"\r\n
\aOverflow\r\n
```

changeGroupSchedule Command

Command	changeGroupSchedule
Shortcut	cgs
Arguments	[group] seconds
Availability	when logged in
Purpose	To schedule automatic, unsolicited, recurring, changeGroupGets. If any changes have occurred when the periodic timer expires, the server will automatically send a change list, as if changeGroupGet has been called. While this mechanism violates the normal client/server relationship, and is not normally recommended (what if too many changes occur too quickly for the control system?), it may be useful in reducing network traffic if the control system is polling a very large number of servers and is looking for changes that need to be recognized quickly. Otherwise, reasonable real-time programming practices on the control system side make this command unnecessary.
Notes	The group name argument is optional, and if it is not included, a Change Group named default Change Group will be used. The schedule can be cancelled by calling changeGroupSchedule with an argument of zero. Only one Change Group can be scheduled.
Response	Something like this: changeGroupSchedule Balcony Speaker Overload Indicators 5 Subsequently, changeGroupGet responses will be returned when Controls in the Change Group have changed.

Usage example

The client issues:

```
changeGroupSchedule "Balcony Speaker Overload Indicators Change
Group" 5\r
```

or

```
cgs "Balcony Speaker Overload Indicators Change Group" 5\r
```

Response

RATC2 responds with the Change Group name and the number of changes that were requested.

```
changeGroupSchedule "Balcony Speaker Overload Indicators Change
Group" 5\r\n
```

Possible Error Messages

```
\aBadArgumentCount\r\n
```

```
\aOverflow\r\n
```

```
\aInvalidChangeGroup "yourBogusGroupName"\r\n
```

Chapter 6

SNMP Control

In This Chapter

Introduction

The MediaMatrix NION series of products supports multiple methods of remote control and monitoring. Simple Network Management Protocol (SNMP) is yet another option that provides a standardized method for monitoring and controlling the NION from third-party systems. SNMP requires just TCP/IP to function.

Software tools

In order to use SNMP, you will need to download a software tool, since most operating systems do not provide one. We recommend *MIB Browser* http://www.mg-soft.com/mgMibBrowserPE.html by MG-Soft for basic browsing and light control. Medium level products like Ipswitch's *What'sUp* http://www.whatsupgold.com/products/ can perform custom actions, send emails, create web-based reports on network conditions, uptime and anything else within your NION Project file. Higher end products, such as Hewlett Packard's OpenView, can do this and more.

Tip: All the screenshots in this topic are from MIB Browser.

What is a MIB?

SNMP is the protocol that your software, as the client, uses to *get* and *set* values on the NioNode. A MIB (Management Information Base) file provides a roadmap that defines the location of the values, and it makes it easier for users to locate them.

The Object Identifier (OID) is the unique location or address of a specific piece of data. You may only want to read it or you may want to replace the current value with a new value. The MIB file allows you to replace or add understandable names to guide you through the otherwise all-numeric OIDs. The MIB file was designed alongside the SNMP agent that runs on our hardware.

MIB files use a standardized plain-text format to define the OIDs available on a product's SNMP agent (the SNMP *server*). The *PEAVEY-NION-NIONODE-MIB-V2.my* file (located in the *NWare\plugins\pion\doc* folder of each installation of NWare) is the *uncompiled* MIB file. We are often asked why we don't just go ahead and compile the MIB for you. The answer is that each SNMP client application utilizes the MIB in a different way. So in every case we've seen, an SNMP client package comes with a special application that can compile any MIB file into the format required for that application's use. Once compiled, you shouldn't have to compile it again (unless the MIB changes). You can load the compiled MIB file into your client program, enter the TCP/IP address of the NioNode that you want to communicate with and start browsing.

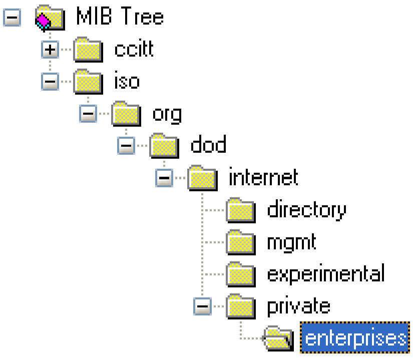
To illustrate how helpful this is, here is a couple of examples. First, you should know that all non-government SNMP OIDs start with these numbers:

```
1.3.6.1.4.1
```

which using a standard RFC1155-SMI MIB (should come with your client) should translate to:

```
iso(1).org(3).dod(6).internet(1).private(4).enterprises(1)
```

It's even unlikely that most would know what the upper set meant without the help of the MIB. This could be arranged in a Explorer-like tree to clarify the concept. The OID numbers and now the English descriptions describe how to get to a specific point in the tree.

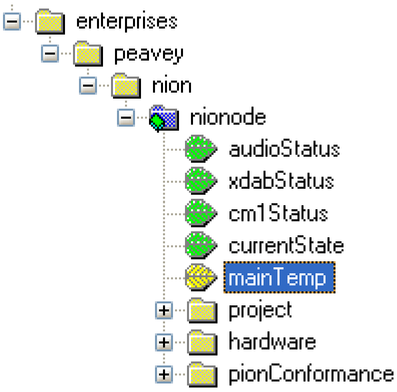


Adding the compiled NioNode MIB, you will add items below `enterprises`. Peavey has registered with IANA the enterprise ID 24603. All our OIDs should always be under that number. An exception is our CobraNet products, which fall under the standard CobraNet MIB under the Peak Audio enterprise ID.

For instance you may want to know the mainboard's temperature. you aren't allowed to set the temperature because the original programmer made a decision to make that control read-only so that only the thermometer could update the value. Here's the OID for mainTemp:

```
so(1).org(3).dod(6).internet(1).private(4).enterprises(1).peavey(24603).nion(1).nionode(1).mainTemp(5)
```

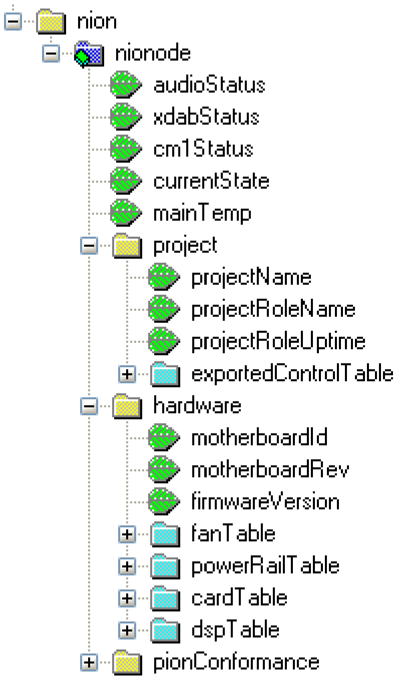
or graphically:



Remember that to access the value, you don't need the descriptions, you just need to know `1.3.6.1.4.1.24603.1.1.5`, but you can see that the description makes everything easier. You can also enter this OID directly into certain programs to have the value polled.

What information can I access?

The picture below shows some of the settings you can access on a NioNode using your MIB browser software.



- nion
  - nionode
    - audioStatus
    - xdabStatus
    - cm1Status
    - currentState
    - mainTemp
    - project
      - projectName
      - projectRoleName
      - projectRoleUptime
      - exportedControlTable
    - hardware
      - motherboardId
      - motherboardRev
      - firmwareVersion
      - fanTable
      - powerRailTable
      - cardTable
      - dspTable
    - pionConformance