Author	S. Lesage
Verified	
Version	1.7.0
Date	2018/08/06
Document	Protocol

VNB Command Framework 1.7.0

For VNB, PMIP-X, Vox@xxx and TSIP

Table of contents

1. Command/Variable Framework

All VNB based products (VNB, PMIP-x, Vox@xxx, TSIP) have a common framework where each firmware component provides a set of commands it can execute and variables to show/change its state.

Firmware Components are basically OS threads that can manage:
- basic I/O hardware like network, serial interface, contacts inputs, relay outputs, audio I/O
- high-level functions like audio routing, audio network streaming, SIP calls, Vox@net client, scheduler, GUI and the command framework itself.

This framework is available for:
- internal use, for example when triggered by a logical input
- external use, for control by an external system via HTTP, TCP, UDP or serial link
- remote control, to control external devices via HTTP, TCP or UDP or serial link

1.1 Variables

Framework components register variables that can be used to control a parameter or read their state. They can be of any type (Boolean, Integer, String).

In particular, **Read-only Logical Variables** show the current state of the components:
- A logical variable has a Boolean value which can be 0 or 1.
- For "normal" variables, 1 usually means active, and 0 inactive.

They can be declared as **State Variables**:
- 1 means OK, 0 means fault.
- They account in the global state in the device (logical AND of all state variables).

You can modify the value of **Writable Variables** with the Framework System Commands.

Refer to `/variablestate.html` web page for a list of all variables of a particular device.

Boolean Variables can be assigned to Logical Outputs.

You can create user-defined Event Commands which are triggered when a Variable changes.

1.1.1 User defined Variables

Via the "User" component, you can create your own variables.
An integer variable will be limited by min and max values.
On reset, all variables can have a defined initial value, or reload the previously set value.

Go to `/variablesconfig.html` web page to configure your variables.

1.2 Command Types

1.2.1 System Commands

System commands are the built-in commands registered by the various components.
These are described further in this document for each component.

Refer to `/commands.html` web page for a list of all system commands of a particular device.

1.2.2 Event Commands

Event commands are also declared by the framework components, but there is no built-in action. They are a place-holder where the user writes a command script. Execution is triggered by the owner component when a particular event happens.

These are described further in this document for each component.

Refer to `/events.html` web page for a list of all events of a particular device and enter your command scripts.

1.2.3 User Commands

You can define your own commands with your own parameters and implement the actions to execute with a command script (TODO: optionally returning your own values)

This is useful in many scenarios:

- you have the same commands to execute in multiple places, with different parameters.
- you want to execute multiple commands within a single key of a PMIP-D / Vox@D.
- you want to define your own protocol for the command line interface (TCP or serial)

Go to `/commands.html` web page to create your own commands.

1.3 Command Line Syntax

The command strings use the URL syntax as described by RFC 1738 for Uniform Resource Locators.

```
scheme://host:port/command?param1=value1&param2=value2&param3=value3…
```

Scheme, host and port are optional and used only in order to control a remote device.
When you specify a host with an IP address, then you must also specify the scheme.

Command is of course mandatory.

Parameters are optional, depending on the command or third party protocol.

Command and parameters are case insensitive.
Parameters values must be "percent encoded". In particular, if you want to use, "&", "%", "=" or "+" characters you must encode them to `%26`, `%25`, `%3D`, or `%2B`. Also, you should not insert spaces, but the parser is tolerant.

1.4 Answer

Responses are formatted using SGML/XML style, terminated by `<CR><LF>`.

"`<ACK/>`" is returned when the command executed successfully.

"`<NACK>error string</NACK>`" is returned when the command failed.
"`<NACK/>`" is returned is there is no detail available.

Commands can also return XML structured data for "get" requests.

1.5 Supported schemes

1.5.1 **Local**

This is the common case, no scheme, no host, no port. The command executes inside the device.

Example, start/stop a SIP call: `call?to=user@192.168.10.171`

1.5.2 **vnb://**

Default port is 12302.
Because TCP needs a connection handshake and is too slow for retries, and because raw UDP is unreliable, this is the preferred protocol to control other VNB Framework based devices (VNB, PMIP-x, Vox@xxx, TSIP).

The device sends the plain text command over UDP. The host must answer within 300 ms.
For reliability, the device retries up to 2 times after a 100 ms time-out.

Example, activate Relay 2 of a remote VNB:
`vnb://192.168.10.159/set?remout2=1`

1.5.3 **tcp://, tcpack://, udp:// and udpack://**

Default port is 12302.
For TCP, the device connects to the specified host, with 300 ms time-out.
For UDP, there is no connection.

The command string is sent without the initial `/` character.

If there are no parameters, the command is *"percent decoded"*, it allows you to send **binary data**: `%xy` sequences are replaced by the character whose code is `xy` in hexadecimal. In this case, if you want to use "?", or "%" characters you must "percent encode" them to `%3f` or `%25`.

For "**ack**" schemes, the device waits for a response from the host for 300 ms.
If the answer begins with the string `"<ACK"` or character `<ACK> = 0x06`, then it returns a successful result.

Time-out and other responses are considered a failure.

Example, write 1 to parameter C001 of an UAPg2/LAPg2/IDA8 using Third Party Protocol:
`udpack://192.168.10.220:19761/%02WC0011%03%5C%0D`

1.5.4 **atpp:// for ATEIS Third Party Protocol (UAPg2, LAPg2, IDA8)**

Default port is 19761.
The command is encapsulated between `<STX>` and `<ETX>` + checksum + `<CR>`, and sent over UDP.

The device waits for a response from the host for 300 ms.
If the answer begins with the character `<ACK> = 0x06`, then it returns a successful result.

For UAPg2, LAPg2 and IDA8 devices, the command must be formatted according to the Ateis Third Party Protocol:
- 1 character for the command type (W to write, I to increment, D to decrement, M for master preset)
- 4 characters for the parameter name
- optional digits for the parameter value

Example, write 3.5 to parameter C005:
```
atpp://192.168.10.215/WC0053.5
```

Example, increment parameter C002 by 4:
```
atpp://192.168.10.215/IC0024
```

1.5.5 **http://**

Default port is 80.
The device connects to the specified host, with 300 ms time-out.
Then it issues a `GET` request, and waits for the response for another 300 ms.

Only the response code is examined (not the content).
If it is `HTTP/1.x 200 OK`, then it returns a successful result.
Time-out and other answers are considered a failure.

Example, start Audio Stream Decoder 1 of a remote PMIP:
```
http://192.168.10.159/decode?channel=1&mode=1&addr=225.1.2.3&port=8000
```

1.5.6 **On the TO DO list**

- com1:
- mailto:

1.6 Command line scripting

1.6.1 Variable substitution

You can use the value of a variable with the following syntax: `${variable_id}`

Example:
On a PMIP-D or Vox@D, you want keys to control the background music of 10 devices by changing the stream address of their audio decoder.

You define a string variable named `stream` and in the associated "streamchanged" event:

```
vnb://192.168.10.170/music?src=udp://&addr=${stream}&dst=1
...
vnb://192.168.10.179/music?src=udp://&addr=${stream}&dst=1
```

Now in the PMIP-D, you can define some keys:

```
Label1=Music 1
Func1=COMMAND
Param1=set?stream=225.0.0.1:8001
...
Label4=Music 4
Func4=COMMAND
Param4=set?stream=225.0.0.4:8004
```

1.6.2 On the TO DO list

- generalization of ${…} syntax for expression evaluation
- conditional execution with `if/else/endif` keywords
- early exit and user defined return value with `return` keyword

2. Command Interfaces

2.1 Internal

Internal command execution happens when a user-defined command script is executed (User Commands or Event Commands, for example, by activation of a logical input).
There is course, no way to know if execution succeeded or to retrieve the answer.

2.2 Serial Interface

On the VNB, Vox@EX/IEX/FDX/AMP/IO and TSIP, you can configure the serial port as command line interface.
Commands strings must be terminated by the `<CR> = 0x0D` character.
Response can be `<ACK> = 0x06`, `<NAK> = 0x15` or XML data.

2.3 UDP Interface (vnb:// protocol)

Always active on port 12302.
You can use a program like UDP Test Tool to experiment.

Protocol Implementation Detail:
When sending a new command, the client must initialize a new socket to use a new source port.
When retrying, it must keep the same source port.

When a receiving a command, the host must parse a list of recently executed commands (source address, source port, command, result), in order to detect a retry.

If a previous identical command is found, it's a retry because the client didn't receive the answer in time, then the host does NOT execute the command again, it reads the original result back.

For a new command, it is executed and the result is stored in the MRU list.

Then the plain text result is sent back to the device over UDP.

2.4 TCP Interface

You must configure it in Vox@net mode. Usual port is 12302. Only 1 connection is served.
Commands strings must be terminated by the `<CR><LF>` sequence.
You can use any telnet program like PuTTY to open a raw interactive session and experiment.

2.5 HTTP Interface

The embedded web server will try to execute any unknown page as a command. It returns HTTP 200 OK with the answer as `text/plain` data, or the classical HTTP error 404 if no command found.
You can experiment with your favorite web browser, try http://192.168.10.159/info for example.

2.6 Graphical User Interface

On the PMIP-D and Vox@D, you can define keys to execute a command when pressed or released, using `COMMAND`, `LATCHCOMMAND` and `COMMANDSEL` functions.
The color of the key will change to indicate the success or failure status.

3. Components

3.1 Framework

3.1.1 Read-only Logical Variables

Variable	Name	Values
Device Global Status	state	global status of the device (logical AND of all state variables)
Device Events	events	currently active event in the device (logical OR of all event variables)
Local Date / Time	time	cf. time format below
GMT Date / Time	gmtime	cf. time format below

3.1.1.1 Date / time format

On a get, the time value is formatted as described in RFC 2822 for Internet Messages.

Example: `get?time`

→ `<state><var id="time" value="Mon, 4 May 2009 16:37:23"/></state>`

On a set, the input parser is more tolerant:

1. `extended RFC 2822:`
`[ day-of-week [ "," ] ] day monthname year hour ":" minute ":" second`

`2. classical:`
`[ day-of-week [ "," ] ] day "/" month "/" year hour ":" minute ":" second`

Examples:
```
set?time=4 May 09 16:37:23
set?time=Mon 4/5/2009 16:37:23
set?time=04/05/09 16:37:23
```

→ `<ACK/>`

3.1.2 **Framework System Commands**

3.1.2.1 Device information

Command: `info`

Parameters: `none`

Examples:
```
<info product="VNB" version="1.1.15" boot="1.2.3" hostname="VNB"
reset="2016-10-03 14:16:37" />

<info product="TSIP" version="1.2.1" boot="1.3.0" hostname="TSIP" />
reset="2016-10-03 14:16:37" />
```

3.1.2.2 Get Variable(s) Value

Command: `get?id1&id2…`

Parameters: id(s) of the variables we want the value

Answer:
```
<state><var id="id1" value="val1" />
<var id="id2" value="val2" />
…
</state>
```

or `<NACK/>` if missing or unknown id

Example: `get?call`

→ `<state><var id="call" value="0" /></state>`

3.1.2.3 Get all State Variables

Command: `getstate`

Parameters: `none`

Example: `<state><var id="remin1state" value="1" /> … </state>`

3.1.2.4 Get Faulty State Variables

Command: `getfaults`

Parameters: `none`

Example:
```
<state><var id="state" value="0" />
<var id="modanconn" value="0" />
<var id="msg1filemissing" value="0" /></state>
```

3.1.2.5 Reset Variable(s) Value

Commands: `reset?id1&id2…`

Parameter	Name	Valid Values
Variable X	idX	id(s) of variables you want to reset
Delay	delay	Optional, 100 to 1000000 ms
Pulse duration	pulse	Optional, 100 to 1000000 ms
Period	period	Optional, pulse+100 to 1000000 ms
Count	count	Optional, 1 to 1000

Answer: `<ACK/>` or `<NACK/>` if unknown variables / invalid value

NB: `delay` parameter is independent of the others.
`pulse` is an optional delay before setting the variable back to its maximal value.
You cannot use `period` without `pulse`.
Then `period` must be greater than `pulse`+100.
You cannot use `count` without `period`.
If `count` is not given, then operation will repeat forever.

Using reset/set/inc/dec/toggle without time control parameters will cancel the previous operation.

3.1.2.6 Set Variable(s) Value

Commands: `set?id1=value1&id2=value2…`

Parameter	Name	Valid Values
Variable X	idX	id(s) and value(s) of the variables you want to set
Delay	delay	Optional, 100 to 1000000 ms
Pulse duration	pulse	Optional, 100 to 1000000 ms
Period	period	Optional, pulse+100 to 1000000 ms
Count	count	Optional, 1 to 1000

Answer: `<ACK/>` or `<NACK/>` if unknown variables / invalid value

NB: `delay` parameter is independent of the others.
`pulse` is an optional delay before setting the variable back to its minimal value.
You cannot use `period` without `pulse`.
Then `period` must be greater than `pulse`+100.
You cannot use `count` without `period`.
If `count` is not given, then operation will repeat forever.

Using reset/set/inc/dec/toggle without time control parameters will cancel the previous operation.

3.1.2.7 Increment Integer Variable(s) Value

Command: `inc?id1&id2=value2…`

Parameter	Name	Valid Values
Variable X	idX	id(s) and value(s) of the integer variables you want to increment (default is by 1)
Delay	delay	Optional, 100 to 1000000 ms
Period	period	Optional, 100 to 1000000 ms
Count	count	Optional, 1 to 1000

Answer: `<ACK/>` or `<NACK/>` if unknown / not integer / upper limit reached / invalid

NB: `delay` parameter is independent of the others.
You cannot use `count` without `period`.

Using reset/set/inc/dec/toggle without time control parameters will cancel the previous operation.

3.1.2.8 Decrement Integer Variable(s) Value

Command: `dec?id1=value1&id2…`

Parameter	Name	Valid Values
Variable X	idX	id(s) and values(s) of the integer variables you want to decrement (default is by 1)
Delay	delay	Optional, 100 to 1000000 ms
Period	period	Optional, 100 to 1000000 ms
Count	count	Optional, 1 to 1000

Answer: `<ACK/>` or `<NACK/>` if unknown / not integer / bottom limit reached / invalid

NB: `delay` parameter is independent of the others.
You cannot use `count` without `period`.

Using reset/set/inc/dec/toggle without time control parameters will cancel the previous operation.

3.1.2.9 Toggle Logical Variable(s) Value

Command: `toggle?id1&id2…`

Parameter	Name	Valid Values
Variable X	idX	id(s) of the integer variables you want to toggle
Delay	delay	Optional, 100 to 1000000 ms
Period	period	Optional, 100 to 1000000 ms
Count	count	Optional, 1 to 1000

Answer: `<ACK/>` or `<NACK/>` if unknown / not integer / invalid value

NB: `delay` parameter is independent of the others.
You cannot use `count` without `period`.
If `count` is not given, then operation will repeat forever.

Using reset/set/inc/dec/toggle without time control parameters will cancel the previous operation.

3.1.2.10 Run/repeat command with delay / period

Command: `run`

Parameter	Name	Valid Values
Command ID	Command	ID of the command you want to run
Delay	delay	Optional, 100 to 1000000 ms
Period	period	Optional, 100 to 1000000 ms
Count	count	Optional, 1 to 1000

Answer: `<ACK/>` or `<NACK/>` if unknown / not integer

NB: `delay` parameter is independent of the others.
You cannot use `count` without `period`.

3.1.2.1 Watch Variable(s) to get change notifications

Command: `watch?id1&id2…`

Parameters: id(s) of the variables you want to watch

Answer: `<ACK/>` or `<NACK/>` if unknown variables

Example: `watch?remin1`

When switching the contact input 1, you will be notified:

```
<state><var id="remin1" value="1" /></state>

<state><var id="remin1" value="0" /></state>
```

Remarks: This works only for the TCP command interface, and is reset when disconnected.

3.1.2.2 Append item to a comma-separated list string

Command: `append?id=value…`

Parameters: id of the string variable you want to append a new item to

Answer: `<ACK/>` or `<NACK/>` if unknown / not string variable

Example:
```
reset?mylist      → mylist is empty
append?mylist=a   → mylist contains "a"
append?mylist=a   → mylist contains "a,a"
append?mylist=b   → mylist contains "a,a,b"
append?mylist=b   → mylist contains "a,a,b,b"
```

3.1.2.3 Append string to build a comma-separated list with unique values

Command: `appendonce?id=value…`

Parameters: id of the string variable you want to append a new unique value to
the value is not added if already present

Answer: `<ACK/>` or `<NACK/>` if unknown / not string variable

Example:
```
reset?mylist                  → mylist is empty
append?mylist=192.168.10.1    → mylist contains "192.168.10.1"
append?mylist=192.168.10.2    → mylist contains "192.168.10.1,192.168.10.2"
append?mylist=192.168.10.1    → mylist contains "192.168.10.1,192.168.10.2"
```

3.1.2.4 Remove item from a comma-separated list string

Command: `remove?id=value…`

Parameters: id of the string variable you want to remove an item from

Answer: `<ACK/>` or `<NACK/>` if unknown / not string variable

Example:
```
reset?mylist                  → mylist is empty
append?mylist=192.168.10.1    → mylist contains "192.168.10.1"
append?mylist=192.168.10.2    → mylist contains "192.168.10.1,192.168.10.2"
remove?mylist=192.168.10.1    → mylist contains "192.168.10.2"
```

3.2 Network

3.2.1 Read-only Logical Variables

Variable	Name	Valid Values
Ethernet link	ethlink	1 for link up, 0 for link-down state variable
IP address	ipaddress	The IP address of the device
IP broadcast address	ipbroadcast	Broadcast address of the device network
NTP state	ntpstate	1 for OK, 0 for time-out optional state variable

3.3 TCP Command Interface

3.3.1 Read-only Logical Variables

Variable	Name	Values
Command interface connection	cmdconn	1 for connected, 0 for disconnected optional state variable

3.4 MODAN over TCP Interface

3.4.1 Read-only Logical Variables

Variable	Name	Values
MODAN interface connection	modanconn	1 for connected, 0 for disconnected optional state variable

3.5 Serial Interface

3.5.1 Read-only Logical Variables

Variable	Name	Values
Serial Gateway Connection	serialconn	1 for connected, 0 for disconnected optional state variable

3.6 Logic I/O

3.6.1 **Read-only Logical Variables**

Variable	**Name**	**Values**
Logical input X value	reminX	0 (inactive), 1 (active) for Logical mode 0 to N for Selector mode with N taps User defined range for Analog mode
Logical input X state	reminXstate	state variable, when input is monitored (also for Selector / Analog)

3.6.2 **Writable Variables**

Variable	**Name**	**Values**
Logical output X value	remoutX	0 (inactive), 1 (active)

Remark: These variables are virtual and do not directly drive the output hardware.

You must assign it to your physical output in the Logic Configuration page.

Example: `set?remout2=1`

→ `<ACK/>`

3.6.3 **Event Commands**

Command Name	**Event**
reminXon	Input X activated (only in Logical mode)
reminXoff	Input X deactivated (only in Logical mode)
reminXisY	Tap Y is selected on Input X (only in Selector mode)
reminXfault	Input X fault (only for monitored inputs or Selector / Analog)
reminXchanged	Input X changed (automatically created in Analog mode)

3.6.4 **System Commands**

3.6.4.1 Control Output

Command: `remout`

Parameter	**Name**	**Valid Values**
Mode	mode	0 (OFF), 1 (ON), 2 (pulse)
Channel	channel	1 to number of outputs
Pulse Duration	time	1 to 10000ms (only for pulse mode)

Answer: `<ACK/>` or `<NACK/>` if unknown parameter or value out of range

Remark: This command controls the value of the virtual "`remoutX`" variable.

You must assign it to your physical output in the Logic Configuration page.

3.7 Audio Stream Decoders

3.7.1 Read-only Logical Variables

Variable	Name	Values
Decoder X active	decXactive	0 (inactive), 1 (active)
Decoder X time-out	decXtimeout	optional state variable, 0 (time-out), 1 (OK)

3.7.2 Writable Variables

Variable	Name	Valid Values
Decoder X level	voloutX	-60 to 0 dB (mute if < -60)

3.7.3 System Commands

Command: `decode`

Parameter	Name	Valid Values
Output Channel	channel	1 or 3, optional, defaults to 1
Mode	mode	0 (stop), 1 (raw UDP), 2 (RTP)
Audio Codec	format	0 (Auto Detect), 1 (MP3), 2 (Speex), 3 (linear PCM) 4 (G.711 A-law), 5 (G.711 mu-law) , 6 (G.722)
Packet Buffering	buf	0 to 16 (min number of packets to start)
Packet Buffering	maxbuf	2 to 16 (max number of packets)
Source Address	addr	any IP address, a.b.c.d
IP Port	port	any IP port, 1 to 65535
Source MRL	src from	network stream MRL proto://address:port

Answer: `<ACK/>` or `<NACK/>` if unknown parameter or out of range value

NB: Parameters not specified are left unchanged (default config/previous command).

If `addr` is multicast, the device sends IGMP join/leave group messages.

In unicast or broadcast mode, `addr` can be used to filter the source, but in general there's no reason to filter, so set `addr=0.0.0.0`.

Examples:

```
decode?from=rtp://225.6.7.8:6000
decode?src=udp://192.168.10.98:8000
decode?src=udp://0.0.0.0:0

decode?channel=2&mode=1&addr=225.1.2.3&port=8000
decode?channel=2&mode=0
```

3.8 Audio Stream Encoders

3.8.1 Read-only Logical Variables

Variable	Name	Values
Encoder X active	encXactive	0 (inactive), 1 (active)
Encoder X audio activity time-out	encXaudioactivity	0 (time-out), 1 (activity OK)

3.8.2 System Commands

Command: `encode`

Parameter	Name	Valid Values
Audio Input channel	channel	1 or 2, optional, defaults to 1
Mode	mode	0 (stop), 1 (raw UDP), 2 (RTP)
Audio Codec	format	1 (MP3), 2 (Speex), 3 (linear PCM) 4 (G.711 A-law), 5 (G.711 mu-law), 6 (G.722)
Sampling rate	freq	MP3: 48000 Speex: 16000 PCM: 8000, 16000, 24000 or 48000 G.711: 8000 G.722: 16000
Bitrate / Quality	bitrate	MP3: 32000 to 320000 bit/s Speex: 6 to 10 quality Other formats: not used
Packet Optimization	fill	0 (1 frame per packet), 1 (fill packet)
Destination Address	addr	one or several IP address(es) a.b.c.d separated by comma or semi-comma.
Destination Port	port	any IP port, 1 to 65535
Time To Live	ttl	1 to 255
Destination MRL	dst to	network Stream MRL proto://address:port
Chime	chime	1 to 100, optional message number to stream as chime before encoding the audio input

Answer: `<ACK/>` or `<NACK/>` if unknown parameter or out of range value

NB: Parameters not specified are left unchanged (default config/previous command).

Examples:

```
encode?to=rtp://225.6.7.8:6000
encode?dst=udp://192.168.10.255:8000&chime=1
encode?dst=udp://0.0.0.0:0      → stop

encode?channel=2&mode=1&format=1&addr=225.1.2.3&port=8000
encode?channel=2&mode=0

encode?channel=1&mode=2&format=2&addr=192.168.10.1,192.168.10.2,
192.168.10.3&port=8010
```

3.9 Audio Engine

3.9.1 Operation

The Audio Engine maintains a list of active "audio routes".

An audio route is composed of:
- Owner: the component which started the route (MODAN, Command, Framework, …)
- Route type: defines how it behaves when interrupted by higher priority sources
- Audio source: analog inputs, user-defined decoder, message, network stream, SIP call
- Audio level
- Priority: only for network streams
- Target outputs: 0 dB lines or amplified speaker "zones"

Whenever a new route is started, modified, or stopped, the Audio Engine updates the list, re-computes the source for all outputs and starts/stops automatic audio stream decoders as needed.

NB: Routes created via the MODAN or TCP command interfaces will automatically be cancelled on disconnection.

3.9.2 Route Types

- *Permanent music*
  The route persists as long as it's not replaced or canceled by routing "none" in target outputs.
  A new permanent route will replace and/or cancel previous routes. It does **NOT** interact with normal routes.

- *Partial Stop on interruption*
  This is the **default** paging mode. When a higher priority source is routed, interrupted outputs are removed from the target outputs. When the source is not used anymore, the route is deleted.

- *Cancel All on interruption*
  This is generally used for **Microphone Call Stations**.
  When a higher priority source is routed and interrupts at least one of the target outputs, then the whole route is canceled and deleted.

- *Suspend on interruption*
  This is the standard mode for **Evacuation Messages**. The route persists as long as it's not explicitly stopped.
  When a higher priority source is routed, it can interrupt one or more target outputs, but the route remains, the source is "suspended" on interrupted outputs.
  When the higher priority source is stopped, then the suspended outputs are active again.

3.9.3 Network streams

When using a network stream source, the Audio Engine automatically allocates/configures/starts/stops the required decoders every time the audio routing table is changed.

To define a networking stream, the audio commands understand the *Media Resource Locator (MRL)* syntax for their "src" parameter:

```
proto://address:port
```

`proto` can be udp or rtp (*TODO tcp and http*)
`address` is an IPv4 address like a.b.c.d, and can be 0.0.0.0 if you don't need to filter the source
`port` is the IP port between 1 and 65535

3.9.4 **Read-only Logical Variables**

Variable	**Name**	**Values**
Global Evacuation State	evac	logical OR of all "outXevac" variables
Output X busy	outXbusy	0 (inactive), 1 (busy)
Output X music	outXmusic	0 (normal source), 1 (music source)
Output X evacuation	outXevac	0 (normal source), 1 (evacuation source)
Input X monitoring status	inXstate	0 (signal missing), 1 (signal OK)
Input X routed	inXrouted	0 (inactive), 1 (used)
Decoder X monitoring status	decXstate	0 (signal missing), 1 (signal OK)
Decoder X routed	decXrouted	0 (inactive), 1 (used)
Any Message routed	msgrouted	0 (no message), 1 (a message is playing)
Message X file missing	msgXfilemissing	dynamic state variables

3.9.5 **Writable Variables**

Master Gain	mastergain	-36 to 18 dB
Music Gain	musicgain	-36 to 18 dB
Normal level	normal	0 (other level), 1 (normal level active)
Night level	night	0 (other level), 1 (night level active)
Attenuated level	attenuated	0 (other level), 1 (attenuated level active)
Loud level	loud	0 (other level), 1 (loud level active)
Security Mode	security	0 (normal), 1 (security)
Output X paging level	outXlevel	-60 to 0 dB
Output X music level	outXmusiclevel	-60 to 0 dB

3.9.6 **Event Commands**

Command Name	**Event**
inXstart	Audio input X activity detected
inXend	Audio input X activity ended
decXstart	Decoder X activity detected
decXend	Decoder X activity ended

3.9.7 System Commands

3.9.7.1 Permanent music routing

Command: `music`

Parameter	Name	Valid Values
Source Channel	src	inX, decX, network stream MRL or "none"
Target Ouputs	dst	comma separated output numbers

Answer: `<ACK/>` or `<NACK/>` if parameter missing or out of range value

Examples:
```
music?src=in2&dst=1,2,3

music?src=none&dst=2

music?src=rtp://0.0.0.0:5000&dst=1

music?src=udp://224.1.2.3:9000&dst=2
```

3.9.7.1 Start audio route

Command: `route`

Parameter	Name	Valid Values
Source Channel	src	inX, decX, msgX or network stream MRL
Target Ouputs	dst	comma separated output numbers
Route Type	type	optional, "stop", "suspend" or "cancel"
Priority	prio	0 to 100, required only for network streams
Chime	chime	1 to 100, optional message number to play as chime before the source

Answer: `<ACK/>` or `<NACK/>` if parameter missing or out of range value

NB: If the type parameter is not specified, it defaults to "stop".

Evacuation messages are automatically started in "suspend" mode.

Examples:
```
route?src=dec2&dst=3

route?src=msg1&dst=1&type=suspend

route?src=in2&dst=1,2&type=cancel&chime=1

route?src=rtp://225.6.7.8:6000&dst=1&prio=90

route?src=udp://0.0.0.0:8000&dst=2,3&type=stop&prio=70
```

3.9.7.2 Stop audio route

Command: `stop`

Parameter	Name	Valid Values
Source Channel	src	inX, decX, msgX or network stream MRL
Target Outputs	dst	comma separated output numbers

Answer: `<ACK/>` or `<NACK/>` if parameter missing or out of range value

NB: If the dst parameter is not specified, it means "all zones".

Examples:
```
stop?src=msg1&dst=2

stop?src=in1

stop?src=rtp://225.6.7.8:6000

stop?src=udp://0.0.0.0:8000&dst=2
```

3.9.7.3 Start streaming message

Command: `stream`

Parameter	Name	Valid Values
Message	msg	message number from 1 to 100
Destination MRL	dst to	network Stream MRL proto://address:port
Mode	mode	0 (Stop), 1 (raw UDP), 2 (RTP)
Destination Address	addr	one or several IP address(es) a.b.c.d separated by comma or semi-comma.
Destination Port	port	any IP port, 1 to 65535
Time To Live	ttl	1 to 255

Answer: `<ACK/>` or `<NACK/>` if parameter missing or out of range value

Examples:
```
stream?msg=1&to=rtp://225.6.7.8:6000
stream?msg=1&dst=udp://192.168.10.255:8000
stream?msg=1&dst=udp://0.0.0.0:0      → stop

stream?msg=1&mode=1&format=1&addr=225.1.2.3&port=8000
stream?msg=1&mode=0

stream?msg=1&mode=2&format=2&addr=192.168.10.1,192.168.10.2,192.
168.10.3&port=8010
```

3.9.7.4 Stop streaming message

Command: `stopstream`

Parameters: none

Answer: `<ACK/>` or `<NACK/>` if parameter missing or out of range value

3.10 TSIP Internal and 100V Lines Monitoring

3.10.1 **Read-only Logical Variables**

Variable	**Name**	**Values**
AC Power Supply	ac	optional state variable
DC Power Supply	dc	optional state variable
Amplifier X Temperature	tempX	temperature in Celcius degrees
Fan X State	fanX	state variables
Monitoring Enabled/Inhibited	monitoring	state variable
All Amplifiers/Lines global status	ampline	state variable
Amplifier/Line X global status	amplineX	state variables
Amplifier X down	ampXdown	dynamic state variables
Amplifier X gain	ampXgain	dynamic state variables
Line X open	lineXopen	dynamic state variables
Line X short-circuit	lineXshort	dynamic state variables
Line X impedance	lineXimp	dynamic state variables
Line X leak	lineXleak	dynamic state variables

3.11 PMIP / Vox@D / Vox@K / Vox@DS Monitoring

3.11.1 Read-only Logical Variables

Variable	Name	Values
Loudspeaker status	spstate	state variable
Microphone status	micstate	state variable

3.12 VNB and Vox@xxx Virtual Monitoring

On the VNB and Vox@xxx, there is no line monitoring.
But we require sometime to report faults of external devices to the vox@net server.
This is achieved by the MODAN component reporting "virtual" monitoring information through variables that the user can freely write (for example, on a contact input event).

3.12.1 Writable Variables

Variable	Name	Values
Amplifier X status	ampX	state variables
Line X status	lineX	state variables

TODO : variables to report deferred and immediate fault

3.13 Vox@net client

3.13.1 Read-only Logical Variables (all devices)

Variable	Name	Values
Vox@net client connection	vnclient	optional state variable
Vox@net system default	vnfault	optional state variable
Vox@net global evacuation	vngloevac	0 (normal), 1 (evacuation)
Vox@net client active	vnactive	0 (inactive), 1 (call active)
Vox@net client 2 active (if having 2 audio inputs)	vnactive2	0 (inactive), 1 (call active)

3.13.2 System Commands (all devices)

3.13.2.1 Play Message

Command: `vnmsgplay`

Parameter	Name	Valid Values
Message	msg	Message file name (without extension)
Target zones	dest	/MxCy,z/MmCn,o,p/…
Priority	prio	1 to 100 or -2, -1, 0 for Low, Medium and High presets C or not specified: default client priority
Chime	chime	not specified: default chime 0 to force NO chime 1 to 8 for selectable chimes

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

Example: `vnmsgplay?msg=blues&prio=50&dest=/M2C1,3,4,5`

3.13.2.2 Stop Message

Command: `vnmsgstop`

Parameter	Name	Valid Values
Target zones	dest	/MxCy,z/MmCn,o,p/…

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

3.13.2.3 Select Music Source

Command: `vnmusicsource`

Parameter	Name	Valid Values
Target zones	dest	/MxCy,z/MmCn,o,p/…
Music source	cli	0 to stop the music 1 to 99 = client number in vox@net configuration

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

3.13.2.4 Enable / Disable / Toggle Scheduler Program

Command: `vnprogram`

Parameter	Name	Valid Values
Program number	prog	1 to 100
Active	act	0 to disable 1 to enable Nothing to toggle

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

Example:
```
vnprogram?prog=1&act=1
vnprogram?prog=2&act=0
vnprogram?prog=3
```

3.13.2.5 Logic Output Control

Command: `vnremoteout`

Parameter	Name	Valid Values
Client number	cli	Optional, 1 to 99 if you want to control a client device
Matrix number	mat	Optional, 1 to 256 if you want to control a matrix device
Output number	out	1 to 16 (depends on the client device)
Mode	mode	0 to force OFF 1 to force ON 2 to pulse 4 to toggle

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

Example:
```
vnremoteout?cli=2&out=1&mode=0
vnremoteout?mat=1&out=1&mode=1
```

3.13.3 **Microphone System Commands (not on PMIP-D and Vox@D)**

On VNB and Vox@xxx, Vox@net can manage 2 clients on the same connection by controlling the 2 decoders / encoders independently through the same TCP command interface.

When command `CMD` addresses client 1, command `CMD2` controls client 2.

3.13.3.1 Start Microphone call

Command: `vnmic / vnmic2`

Parameter	Name	Valid Values
Target zones	dest	/MxCy,z/MmCn,o,p/…
Priority	prio	1 to 100 or -2, -1, 0 for Low, Medium and High presets C or not specified: default client priority

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

Example: `vnmic?dest=/M1C1,3/M2C2`

3.13.3.2 Start Chime call

Command: `vngong / vngong2`

Parameter	Name	Valid Values
Target zones	dest	/MxCy,z/MmCn,o,p/…
Priority	prio	1 to 100 or -2, -1, 0 for Low, Medium and High presets
Chime	chime	not specified: default chime with client priority 1 to 8 for selectable chimes and associated priority -2, -1, 0 for default chime with Low, Medium or High priority preset.

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server or missing parameter

3.13.3.3 Stop call

Command: `vnoff / vnoff2`

Parameters: `none`

Answer: `<ACK/>` if OK
`<NACK/>` if not connected to server

3.14 Scheduler

3.14.1 **Writable Logical Variables**

Variable	**Name**	**Values**
Program Enabled	"prgid"enabled	0/1 to disable/enable the program For each program, you have a Boolean variable whose name is starting with the program ID and followed by "enabled"

3.14.2 **Event Commands**

Command Name	**Event**
"prgid"	For each program, you have an event with the same ID where you place the commands to execute when the program is triggered.

3.14.3 **System Commands**

3.14.3.1 Start program

Command: `startprogram?id1&id2 …`

Parameters: id(s) of the programs you want to start

Answer: `<ACK/>` or `<NACK/>` if unknown/disabled/automatic program

3.15 SIP User Agent

3.15.1 Read-only Logical Variables

Variable	Name	Values
Account "user" active	sip"user"active	"user" account is currently active
Account "user" Ringing	sip"user"ringing	"user" account is ringing from incoming call
Account "user" Incoming	sip"user"incoming	"user" account current call is incoming
Account "user" Outgoing	sip"user"outgoing	"user" account current call is outgoing

3.15.2 Event Commands

Command Name	Event
sip"user"start	"user" account activity starts
sip"user"stop	"user" account activity stops
sip"user"ringingstart	"user" account ringing starts
sip"user"ringingstop	"user" account ringing stops
sip"user"incomingstart	"user" account incoming call starts
sip"user"incomingstop	"user" account incoming call stops
sip"user"outgoingstart	"user" account outgoing call starts
sip"user"outgoingstop	"user" account outgoing call stops

3.15.3 System Commands

3.15.3.1 Call

Command: `call?from=account&to=user@ipaddress`

Parameter	Name	Valid Values
Target account	to	user@ipaddress
Optional Local account	from	any locally defined user account id

Answer: `<ACK/>` or `<NACK/>` if unknown account/already busy/invalid target …

3.15.3.1 Answer

Command: `intercom?from=account`

Parameter	Name	Valid Values
Optional Local account	from	any locally defined user account id

Answer: `<ACK/>` or `<NACK/>` if unknown account/not ringing …

3.15.3.2 Hang-up

Command: `hangup?from=account`

Parameter	Name	Valid Values
Optional Local account	from	any locally defined user account id

Answer: `<ACK/>` or `<NACK/>` if unknown account/not busy …

3.15.3.1 All-In-One call/answer/hang-up

Command: `intercom?from=account&to=user@ipaddress`

Parameter	Name	Valid Values
Target account	to	user@ipaddress
Optional Local account	from	any locally defined user account id

Answer: `<ACK/>` or `<NACK/>` if unknown account/invalid target…

This command is useful when you want only one button to do everything.
It will behave according to the local user current state:

- if in active call, then hang-up
- if ringing to an incoming call from any user, then answer
- if not active, then call the specified target