



Wednesday, May 19, 2010

Dovado API Specification

The DOVADO API interface allows you to create your own interface to the DOVADO UMR in order to steer Home Automation, and manage SMS messaging. Listed below are instructions on managing the API.

Enabling the API:

LAN Access:

***SYSTEM-> REMOTE MANAGEMENT-> ENABLE LAN MANAGEMENT-> API OVER LAN**

WAN Access:

***SYSTEM-> REMOTE MANAGEMENT-> ENABLE WAN MANAGEMENT**

Enter the IP address or hostname of the trusted client or check 'all' to enable WAN access for any client. Check the API, HOME and SMS -checkboxes for Home Automation and SMS access for the client.

Don't forget to reboot the router after activating the API.

Connecting to the API:

The API operates over telnet on port 6435.

To connect you can, for example, type the following in a console:

telnet 192.168.0.1 6435 (or change 192.168.0.1 to the IP of your UMR)

If the connection was successful you will receive something similar to:

This is version X.Y of the API

SMS is enabled and HOMEAUTOMATION is enabled for you

Followed by a '>>'-prompt

Logging in to the API:

The API uses the Username and Password as specified in the Web GUI.

To log in as user admin, type:

user admin

When prompted for a password, enter it using the following syntax:

pass password

(The password is the same as of the UMR Web GUI)

If the login fails, control and reenter your username and password until access is granted.

Using the API:

To display information about the GUI functions and syntax, type ***help***.

Home Automation

Listing configured aliases:

ts aliases

Returns a list of the current aliases and their settings

Adding a new alias:

The current protocols to choose from are:

NEXA, WAVEMAN, SARTANO, IKEA, NEXASELFLEARNING, NEXADIMMER, ARCTEC, RISINGSUN, BRATECK, GEMBIRD, WOL (Wake On LAN)

Always use CAPITAL letters for the protocol!

Examples:

Adding a NEXA alias: `ts add <alias> <protocol> <house> <channel>`

Example: ***ts add myfridge NEXA A 1***

Adding an IKEA alias: `ts add <alias> <protocol> <system> <device>`

Example: ***ts add mylamp IKEA 1 1***

Adding a SARTANO alias: `ts add <alias> <protocol> <channel>`

Example: ***ts add myfan SARTANO 01001001001***

Adding a Wake On LAN alias: `ts add <alias> <protocol> <mac>`

Example: ***ts add mycomputer WOL 11:22:33:44:55:66***

Adding a Gambird alias: `ts add <alias> <protocol> <outlet>`

Example: ***ts add mygembird GEMBIRD 2***

If the alias was successfully added you can see it when listing the aliases. If the add was unsuccessful then review your syntax carefully.

Removing an alias:

Aliases are removed using the following syntax:

ts remove <alias>

Example:

ts remove mylamp

Turn aliases on or off:

Aliases are turned on or off using the following syntax:

ts turn <alias> <on/off>

Example turn on: ***ts turn mylamp on***

Example turn off: ***ts turn mylamp off***

If you cannot turn your alias on/off:

- Check the hardware settings on your device
- Check the configurations on your alises

Groups:

Groups can be useful for controlling multiple devices with a single command. When configured, a group can be turned on/off using the regular 'ts turn' -command (dimming not supported).

There will always be one group, 'all', available. This group contains all configured aliases.

When executing group commands, they are executed in a random sequential fashion per device.

Adding:

ts add_group [alias] [list_of_device_aliases] (the list is a space separated list of devices)

Example: **ts add_group livingroom myfridge mylamp myotherlamp**

Removing:

ts remove_group [alias]

Example: **ts remove_group livingroom**

Listing:

ts groups

Dimming:

If an alias is configured as a dimmer these extended commands are available:

Dim the device to XX%:

ts dim [alias] [0...100]

Example: **ts dim mydimmer 70**

Turn the device off and dim to XX%:

ts dims [alias] [0...100]

Example: **ts dims mydimmer 60**

Dim from start to stop level over XX minutes:

ts dimot [alias] [Start level 0...100] [Stop level 0...100] [Duration xx min]

Example: **ts dimot mydimmer 20 80 5**

SMS

All SMS are handled using the PDU (protocol description unit) mode. To view the clear-text contents of a PDU you need to convert it either on your own or, preferably, using a third-party software

<http://stud.usv.ro/~amurariu/SMS%20and%20PDU%20format.htm>

Listing the number of PDUs:

You can list the number of PDUs on your UMR using: **sms list**

The first digit of the response shows the current number of unread PDU's and the second digit the total amount in the inbox.

Receiving a PDU/Reading an SMS:

You can list all the current PDUs in your inbox using: **sms recv**

This will list all PDUs using the format ID:PDU where ID is a unique internal descriptor of that PDU (used for removing PDUs). Conversion is needed to see the actual contents of the PDU.

Sending an SMS:

To send a SMS, you first have to make a PDU of it. Once you have the PDU, use the following syntax: **sms send <pdu>**

Example: **sms send 07910447946400F011000A9270042079330000AA0CC337392C2F83A6CD292804**

Removing a PDU:

First you need the ID of the PDU you want to remove. It can be obtained using the **recv**-command. Then to remove a PDU, type:

sms del ID

Example: **sms del 2**

Misc

Services available in current session:

To display which services that are available in current session, execute the command:

services

It will respond with the services (Home Automation and SMS) that are currently available

Example:

HOMEAUTOMATION=enabled

SMS=enabled

Router info:

To display information about the router, execute the command:

info

It will respond with a list of parameters, for example:

Firmware_Revision:4.2.0

API_version:1.1

PRODUCT_NAME=Dovado UMR

SIGNAL_STRENGTH=-

TRAFFIC_MODEM_TX=0

TRAFFIC_MODEM_RX=0

TRAFFIC_WAN_TX=0

TRAFFIC_WAN_RX=0

CONNECTION=modem

MODEM_STATUS=DISCONNECTED

EXTERNAL_IP=-

DATE=2010-04-10

TIME=08:11

GPS_TYPE=FIXED

GPS_LAT=59.2222

GPS_LONG=18.1111

SUNRISE=05:48

SUNSET=19:50

SMS_UNREAD=0

SMS_TOTAL=0

CONNECTED_DEVICES=TELLSTICK

API Coding

The API can easily be extended with a custom front-end. When implementing a front-end, mind the following:

- The router might be behind a high-latency connection. Set a high timeout for your connection.
- When logging in to the API, the password must be encoded using ISO-8859-1
- When executing a command (for example ts aliases), each line of output from the API is terminated with LF (Line Feed, ASCII character 10d) and the last line of output from the command is ETB (End of Transmission Block, ASCII character 23d).
- Since there can only be one instance of the API running on the Router, a session that has been inactive for more than 60 seconds will be dropped in favor of a new session.

<- End of document ->