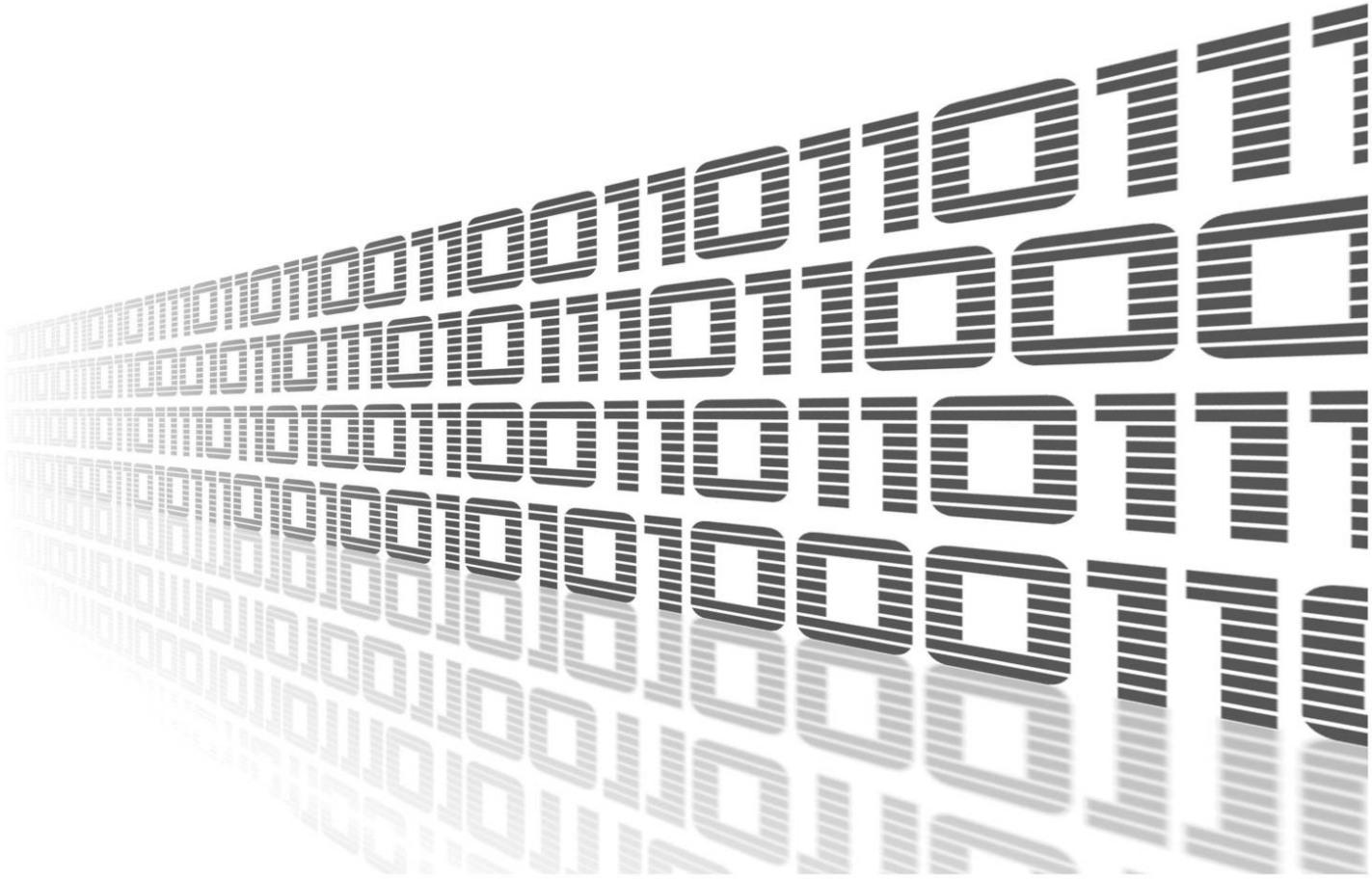
ADVANTECH

RouterApp

Node.js



Advantech Czech s.r.o., Sokolska 71, 562 04 Usti nad Orlici, Czech Republic
Document No. APP-0080-EN, revision from 12th October, 2023.

Used symbols

Danger – Information regarding user safety or potential damage to the router.

Attention – Problems that can arise in specific situations.

Information – Useful tips or information of special interest.

Example – Example of function, command or script.

Contents

List of Figures

List of Tables

1. Changelog

1.1 Node.js Changelog

v1.0.0 (2017-10-02)

- First release.

v1.1.0 (2017-11-08)

- Updated to Node.js 8.9.1.

v1.2.0 (2018-02-18)

- Added support for logging to file with rotating.

v1.2.1 (2018-08-10)

- Updated to Node.js 8.11.1.

v2.0.0 (2020-02-21)

- Updated to Node.js 10.15.3 and ffi 2.3.0.
- Optimized installing nodes files to reduce size.
- Prepared for new GCC 7.4.
- Prepared for new kernel 4.14.
- Prepared for V4 platform.
- Added the custom node "router".
- Set a default path for searching nodes to /usr/lib/node_modules.

v2.1.0 (2021-05-06)

- Updated to Node.js 10.23.1.
- Moved license information from Node-RED module.

v16.14.2 (2022-03-18)

- Updated to Node.js 16.14.2 with npm 8.5.0.

v16.15.0 (2022-05-10)

- Added a object for working with the router configuration to the router node.
- Updated to Node.js 16.15.0 with npm 8.5.5.
- Fixed login on FW 6.3.5.

v16.17.0 (2022-08-25)

- Updated to Node.js 16.17.0 with npm 8.15.0.
- Added property productModel to the router node.

v18.15.0 (2023-04-06)

- Updated to Node.js 18.15.0 with npm 9.5.0.
- Removed obsolte useless node "when".

2. Node.js Router App

Router app *Node.js* is not contained in the standard router firmware. Uploading of this router app is described in the Configuration manual (see Chapter Related Documents). **This router app is only compatible with v3 and v4 platform routers!**

2.1 Introduction

The *Node.js* node is a proprietary server-side JavaScript runtime environment node available for Advantech cellular routers. This node is used by Advantech modules written in JavaScript, but can be used by any other third-party JavaScript application for routers administration and maintenance.

Router module contains this nodes addition to buil-in nodes:

- **node-authenticate-pam** – asynchronous PAM authentication for NodeJS,
- **router node** – a proprietary node for Advantech's cellular routers described in this document in detail.

2.2 Building the Custom Nodes

An official way how to build and install a node is using `npm` command. However, there are some limitation as Advantech routers are embedded devices without a full Linux OS and with specialized hardware. You can install `npm` Router App to the router and use it in the common way, or prepare nodes with npm tool on your PC and then copy them to the router. But it is not possible to install all nodes you can find in the npm repository.

For more details see: `https://icr.advantech.cz/products/software/user-modules#node-red` in the chapter 4.5 of Node-RED Application Note.

3. Router Node

This part of the document is dedicated especially to programmers.

Router node (named "router") provides access to router specific functions and hardware. You can load the *Node.js* node in your code by require("router"), for example:

```
var r = require("router");
```

We will use the *r* variable from this example to access all the properties in the next examples in this notes.

Simple Example of Router Node Use

In the next figure is an example of loading the *Node.js* node.

```
# NODE_PATH=/usr/lib/node_modules/ node
> var r = require("router");
undefined
> console.log("Welcome to Advantech router " + r.productName);
Welcome to Advantech router ICR-323x
undefined
>
```

Figure 1: Loading the Node.js Node

3.1 Node Properties

3.1.1 productName

Read-only string variable loaded with router's product name. Example of usage:

```
console.log(r.productName);
```

Output: *SPECTRE-v3T-LTE*

3.1.2 productModel

Read-only string variable loaded with router's model indication. Example of usage:

```
console.log(r.productModel);
```

Output: *ICR-3201W*

3.1.3 productRevision

Read-only string variable loaded with router's product revision number. Example of usage:
`console.log(r.productRevision);`

Output: *1.0*

3.1.4 platformCode

Read-only string variable loaded with router's platform code. It is supported by routers of v3 and v4 production patform. Example of usage:

```
console.log(r.platformCode);
```

Output: V3

3.1.5 serialNumber

Read-only string variable loaded with router's serial number. Example of usage:

```
console.log(r.serialNumber);
```

Output: ACZ1100000322054

3.1.6 firmwareVersion

Read-only string variable loaded with router's firmware version. Example of usage:

```
console.log(r.firmwareVersion);
```

Output: 6.2.1 (2019-10-16)

3.1.7 RTCBatteryOK

Read-only boolean variable loaded with router's RTC battery state. True means OK, false means bad. Example of usage:

```
console.log(r.RTCBatteryOK);
```

Output: true

3.1.8 powerSupply

Read-only decimal number variable loaded with router's power supply voltage. Example of usage:

```
console.log(r.powerSupply + " V");
```

Output: 11.701 V

3.1.9 temperature

Read-only integer number variable loaded with router's internal temperature in Celsius degrees. Example of usage:

```
console.log(r.temperature + " °C");
```

Output: 39 °C

3.1.10 usrLED

Write-only boolean variable for control router's "USR" LED. Example of usage:

```
r.usrLED = true;
```

Sets USR LED to ON (lighting).

3.1.11 bIn

Read-only array with values on router's binary inputs. Array has the items related to number of the binary inputs. E.g. the router has BIN0 and BIN1 so array has valid indexes 0 and 1. The array items can have values 0 or 1. Example of usage:

```
console.log("The secondary binary input: " + r.bIn[1]);
```

Output: The secondary binary input: 0

3.1.12 bOut

Array related to router's binary outputs. It is similar as B_IN but you can also write values. Writen value change output state. Example of usage:

```
console.log(r.bOut[0]);
```

Output: 1

```
r.bOut[0] = 0;
```

Sets the first binary output to 0.

3.1.13 XBus

Object for working with X Bus. X Bus is a proprietary bus for communication between processes. E.g. you can subscribe informations which network interface go up/down or SMS from a mwan daemon. You can also send/subscribe your own topics between your applications.

```
XBus.publish(topic, payload, store=false)
```

Sends message with topic String and payload String to X Bus. Example of usage:

```
r.xBus.publish("watchdog/proc/myapp", "Timeout: 300");
```

Sends to the system watch request to watch your "myapp" application. The application must send this message regulary no later then period defined in the previous message (300 s in this example). Timeout 0 stops watching.

```
XBus.subscribe(topic, callback)
```

Subscribes to get messages with topic. Example of usage:

Function:
```
xbus.subscribe("status/mobile/mwan0", (msg) => {console.log(msg.payload);});
```

Asynchronous output:
```
Registration: Home Network
```

```
Technology: LTE
Signal-Strength: -88 dBm
Signal-Quality: -8 dB
```

```
XBus.unsubscribe(topic)
```

Unsubscribe from topic. Example of usage:

```
r.XBus.unsubscribe(id);
```

Stops receiving info abour registration to network from previous example.

```
XBus.list()
```

Lists stored messages. Example of usage:

```
r.XBus.list();
```

Output:

```
[ 'iface/ipv4/mwan0/config',
  'iface/ipv4/mwan0/running',
  'iface/ipv4/mwan1/config',
  'iface/ipv4/mwan1/running',
  'status/mobile/mwan0',
  'status/mobile/mwan1',
  'watchdog/proc/bard',
  'watchdog/proc/bard6',
  'watchdog/proc/mwan1d',
  'watchdog/proc/mwan2d',
  'watchdog/proc/mwanxd' ]
```

XBus.read(topic)

Read stored message from XBus. Example of usage:

```
r.XBus.read("iface/ipv4/mwan0/config");
```

Output:

```
Up: 1
Iface: usb0
Address: 10.184.131.221
Gateway: 192.168.253.254
DNS1: 217.77.165.211
DNS2: 217.77.165.81
```

3.1.14 configuration

Object containing the router configuration. User can read a configuration item by geting a object property and write a configuration item by setting a object property. The object keys are the same as configuration keys as in the setting files. It is possible to look for a requested key name in related setting file. The firmware configurations are placed in the `/etc/settings.*` files. Router App's configuration are placed in the `/opt/*/etc/settings` files. The Router Report (Web UI: Status / System Log / Save Report) contains a full list of the current configuration and may be it is the easest way how to find the requested configuration key.

If a given key does not exist a read value is undefined and a written value cause exception (in strict mode). It is not possible to add a new non-existing configuration item, only to modify an existing one. The all configuration values are treated as strings. If user need to work with a different type he must convert it himself. Node does not perform any value validation. The user is responsible for sending the correct values. Examples:

```
console.log("Current router AP SSID is " + r.configuration["WIFI_AP_SSID"]);
```

For WIFI_AP_SSID=ROUTER_AP in /etc/settings.wifi_ap (or rather in the SSID field in the WiFi • Access Point 1 form) output will be:

```
Current router AP SSID is ROUTER_AP
```

An example how to set a configuration value:

```
r.configuration["ETH_IPADDR"]="192.168.99.1"
```

Changes the IP addres on eth0 interface

NOTE: A new configuration is only written. If user wants it to apply to the running environment restarting the router or the related service is necessary. For example above it is possible to use the following shell command:

```
/etc/init.d/eth restart
```

4. Related Documents

[1] Router apps: **icr.advantech.cz/user-modules**
[2] JS Foundation: **https://nodered.org/**

You can obtain product-related documents on *Engineering Portal* at *icr.advantech.cz* address.

To get your router's *Quick Start Guide*, *User Manual*, *Configuration Manual*, or *Firmware* go to the *Router Models* page, find the required model, and switch to the *Manuals* or *Firmware* tab, respectively.

The *Router Apps* installation packages and manuals are available on the *Router Apps* page.

For the *Development Documents*, go to the *DevZone* page.