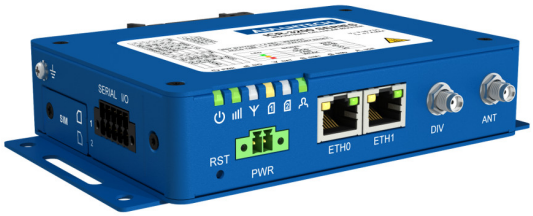

ADVANTECH RouterApp Node.js User Guide

RouterApp

ADVANTECH

User Module

Node.js

APPLICATION NOTE

Contents

Used symbols

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

Attention – Problems that may arise in specific situations.

Information or notice – Useful tips or information of special interest.

Example – Example of function, command or script.

Advantech Czech s.r.o., Sokolska 71, 562 04 Usti nad Orlici, Czech Republic
Document No. APP-0080-EN was revised on May 7, 2021. Released in the Czech Republic.

Node.js User Module

Web Interface

Once the installation of the module is complete, the module's GUI can be invoked by clicking the module name on the User modules page of the router's web interface. The left part of this GUI contains a menu with a General menu section. The general menu section contains only the Licenses containing the list of all licenses for Node.js itself and also related Router Application and Return item, which switches back from the module's web page to the router's web configuration pages. The main menu of the module's GUI is shown on Figure 2.

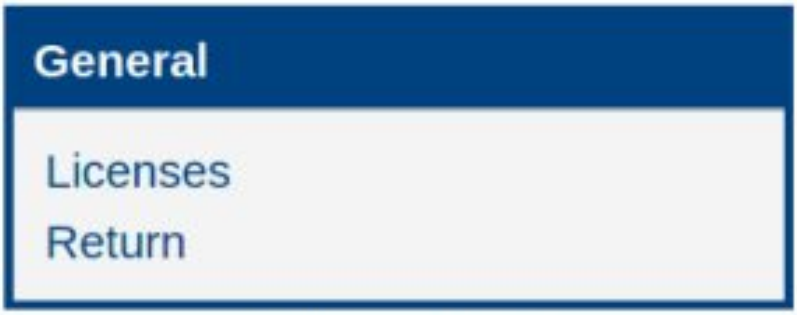

Figure 1: Menu

Introduction

The Node.js user module is not part of the router's firmware. It can be downloaded from icr.advantech.cz/user-modules. The installation process for the user modules is described in the Configuration Manual (see [1], [2], [3] and [4]). This user module is only compatible with v3 and v4 platform routers!
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 the addition of this node to build-in nodes:

- **node-authenticate-pam** – asynchronous PAM authentication for NodeJS,
- **when.js** – Promises/A+ and when() implementation, including a complete ES6 Promise shim,
- **router node –** a proprietary node for Advantech's cellular routers described in this document in detail.

Building the Custom Nodes

An official way how to build and install a node is using npm command. However, it is not possible to find it on our routers as the router is embedded device with limited resources and some nodes require a complex building environment and high performance because of other languages than JavaScript.
Fortunately, it is easy to prepare a node on a PC with Linux and then copy it to the router.
For more details see https://icr.advantech.cz/support/faq/detail/building-the-custom-nodes-fornode-js-node-red.

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 note.
Simple Example of Router Node Use
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

Node Properties

2.1.1 **productName**
Read-only string variable loaded with router's product name. Example of usage:
console.log(r.productName);
Output: SPECTRE-v3T-LTE
2.1.2 **platformCode**
Read-only string variable loaded with router's platform code. It is supported by routers of

v3 and v4 production platforms. Example of usage:
console.log(r.platformCode);
Output: V3

2.1.3 **serialNumber**

Read-only string variable loaded with router's serial number. Example of usage:
console.log(r.serialNumber);
Output: ACZ1100000322054

2.1.4 **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)

2.1.5 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

2.1.6 **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

2.1.7 **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

2.1.8 **usrLED**

Write-only boolean variable for control router's "USR" LED. Example of usage:
r.usrLED = true;
Sets USR LED to ON (lighting).

2.1.9 **bIn**

Read-only array with values on router's binary inputs. The array has items related to a number of binary inputs. E.g. the router has BIN0 and BIN1 so the 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

2.1.10 **bOut**

Array related to router's binary outputs. It is similar to B_IN but you can also write values.
Written 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.

2.1.11 **XBus**

The object for working with X Bus. X Bus is a proprietary bus for communication between processes.
E.g. you can subscribe information which network interface go up/down or SMS from a man 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 regularly no later than the 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 the topic. Example of usage:
r.XBus.unsubscribe(id);
Stops receiving info about registration to network from the 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 messages from XBus. Example of usage:
r.XBus.read('face/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

Related Documents

[1] Advantech Czech: SmartStart Configuration Manual (MAN-0022-EN)
[2] Advantech Czech: SmartFlex Configuration Manual (MAN-0023-EN)
[3] Advantech Czech: SmartMotion Configuration Manual (MAN-0024-EN)
[4] Advantech Czech: ICR-3200 Configuration Manual (MAN-0042-EN)
[5] User Modules: **icr.advantech.cz/user-modules**
[6] JS Foundation: **https://nodered.org/**


[EP] Product-related documents and applications can be obtained on Engineering Portal at **icr.advantech.cz** address.

Documents / Resources

	**ADVANTECH RouterApp Node.js** [pdf] User Guide ADVANTECH, RouterApp, Node.js