Development Container

Pushing Performance

People | Power | Partnership

HARTING Node.js Environment for HAIIC MICA

HARTING IT Software Development
Marienwerder Str. 3, 32339 Espelkamp, Germany
Phone: +49 5572 47-97300, Fax: +49 5772 47-482
mica@HARTING.com

Node.js Environment v1.3. for HAIIC MICA Guide

Content

1. Node.js environment container basics

1.1. Overview

Node.js is an open-source project which is hosted by the Node.js Foundation[1] and a collaborative project of the Linux foundation[2]. It implements a light weighted JavaScript runtime built on the Chrome V8 engine and provides an event-driven architecture and a non-blocking I/O API designed to optimize an application's throughput and scalability for real-time Web applications. The built-in library allows Node.js applications to act as a standalone Web server. Node.js projects can be managed and hosted by the npm-package manager, which claims to provide the largest ecosystem of open-source libraries in the world.

Note that the Node.js Environment Container is currently based on Node.js v6.9.1 LTS. For more information about Node.js and NPM visit https://nodejs.org/en/ and https://www.npmjs.com/.

1.2. Installation of the Node.js container

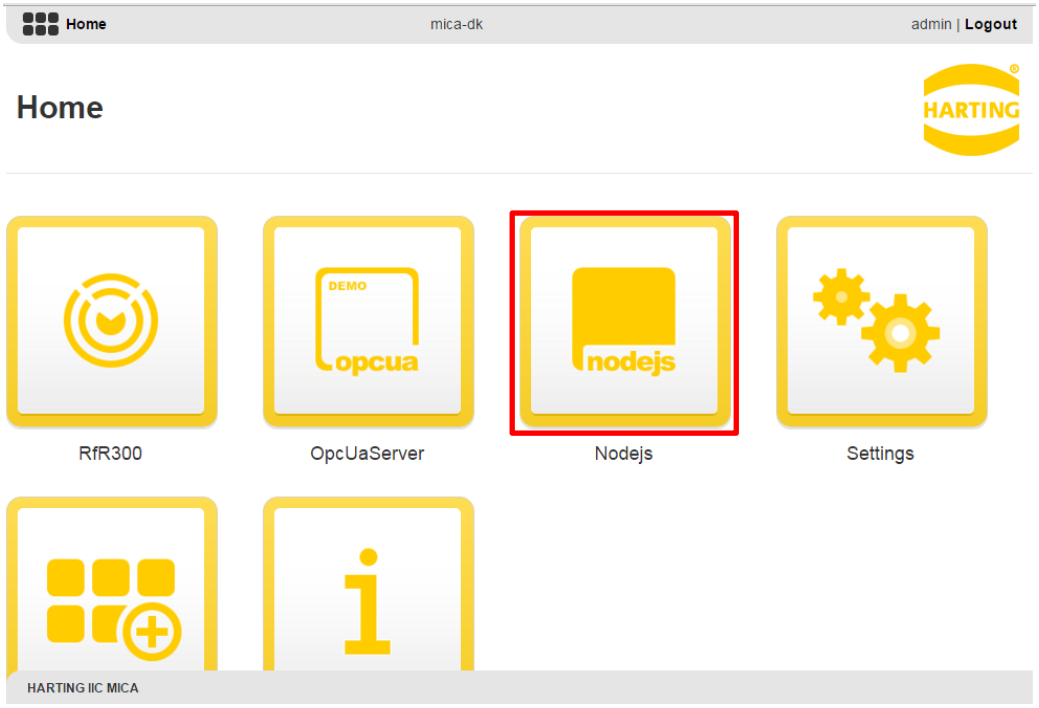

Figure 1: IIC MICA home screen including a (highlighted) Node.js Environment Container

[1] *Node.js Foundation (https://nodejs.org/en/foundation/, Retrieved: 2016/01/22)*

[2] *Linux Foundation Collaborative Projects (http://collabprojects.linuxfoundation.org/, Retreived: 2016701/22)*

The installation and configuration routine of the Node.js Environment Container follows the standard routine as provided by the IIC MICA and can be found in the “MICA Programming Guide”.

2. User-interface of the Node.js container

2.1. Overview of the user interface

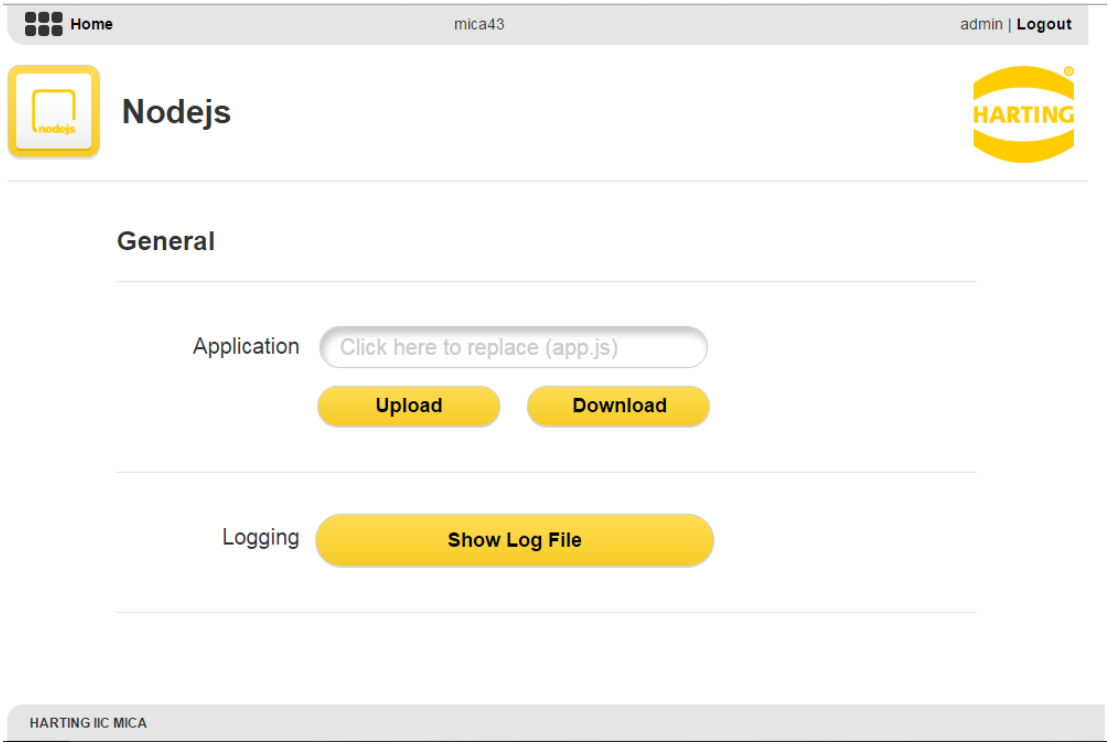

Figure 2: Node.js Environment Container user-interface

The user-interface of the Node.js Environment Container consists of a single section “General” including the entries “Application” and “Logging”, which can be used to import/export an application to/from the container and to show the application logs. The user-interface structure can be described as follows:

1.) **Application file text field**: The text field can be used to set the js-application script you want to upload to the container.

2.) **Upload**: The upload button engages the upload of the js-application file that has been specified in **1.)**

3.) **Download**: The download button engages the download of the current active js-application to your file system.

4.) **Show Log File**: The log file button lets you enter the log window as shown in Figure 5.

2.2. Uploading a Node.js application script

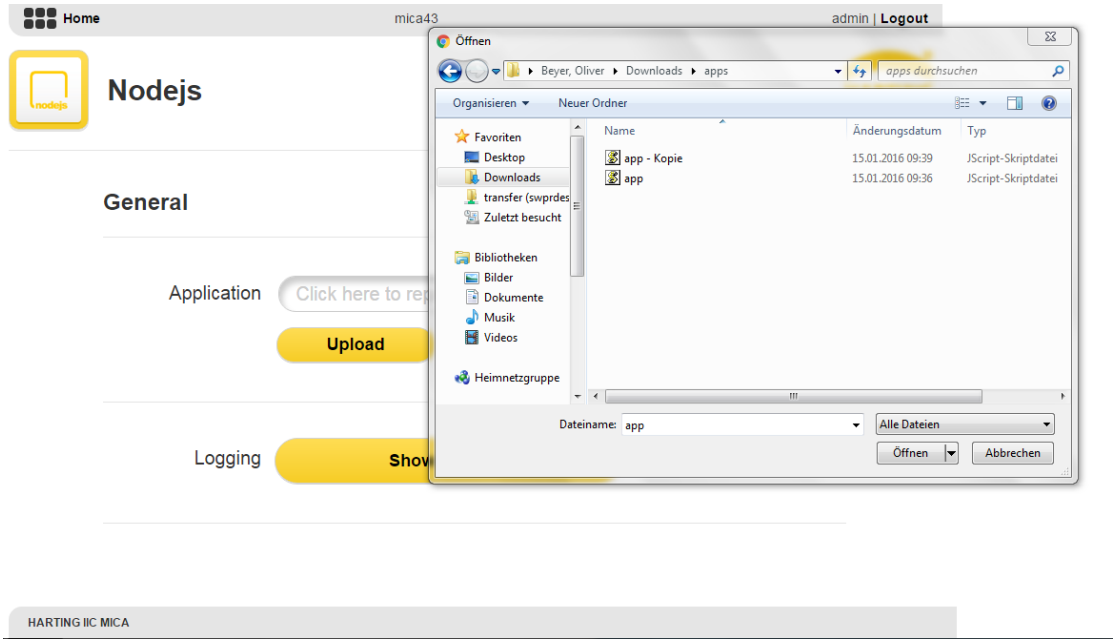

Figure 3: Uploading an application to the Node.js Environment

In order to upload a new application to the container you have to either click the application file text field (see Section 2.1) and select a js-application script as shown in Figure 3, or you simply drag & drop the file into the text field. After selecting a file the file name is shown in the Application file text field.

When pressing the button “Upload” the selected file will be sent to the container and stored internally. After the transmission ended a symbol to the right side of the application file text file will indicate if the transmission was successful or failed, as shown in Figure 4. The container will furthermore stop the application to be replaced and start your new application as soon as the file transmission ended. Note that the uploaded application will be renamed to `app.js` and will also automatically be started after the container itself started.

Caution: You can only run one application at a time when using the user interface as with every upload the old application will be replaced and started. However, you can alternatively log into the container via SSH (as described in the “MICA Programming Guide”) and run several Node.js applications at the same time.

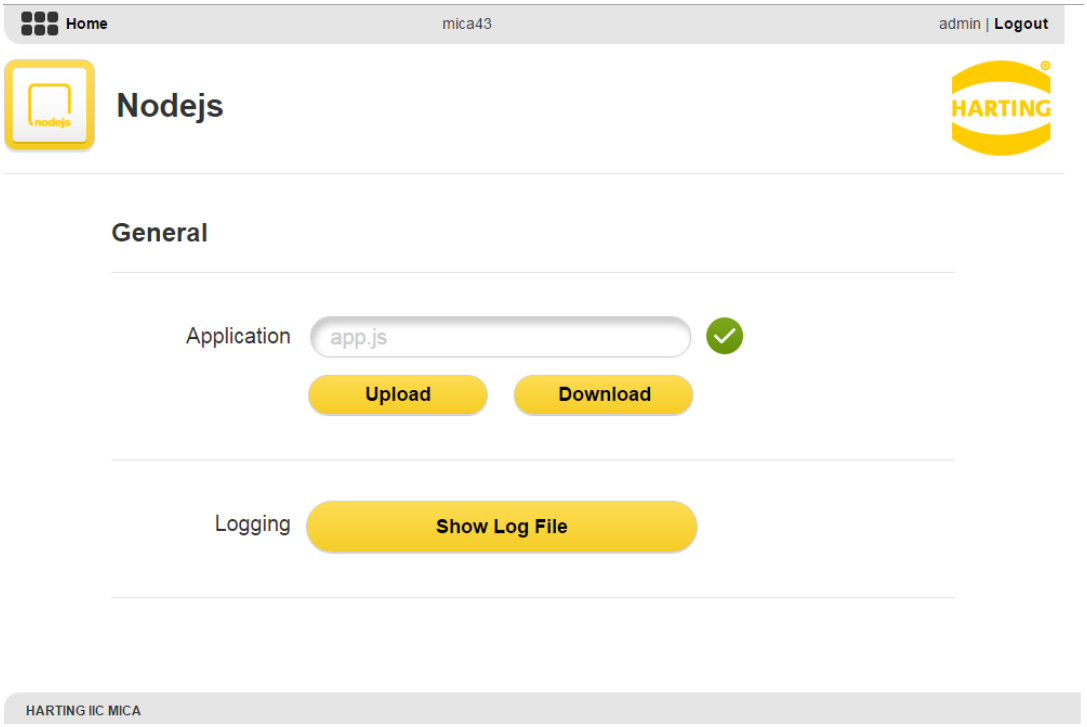

Figure 4: Successful upload of an application to the Node.js Environment

2.3. Downloading the active Node.js application script

You can download the current active Node.js script by simply clicking the download button. After having downloaded the script you can open the file in a simple text editor such as “WordPad” (Windows) and “TextEdit” (OS X) or in a development IDE such as “Visual Studio” (Windows) and “Xcode” (OS X), in order to see/modify the application and re-upload it as described in Section 2.2.

2.4. Using the application logger

Log files of the active application can be inspected in the logger window (see Figure 5) by clicking the “Show Log File”. In order to use the application logger your application needs to write its log information to the following file:

```
/var/log/nodeapp.log
```

For more information about how to write log information in Node.js see the default js-application that is delivered with every new Node.js Environment Container.

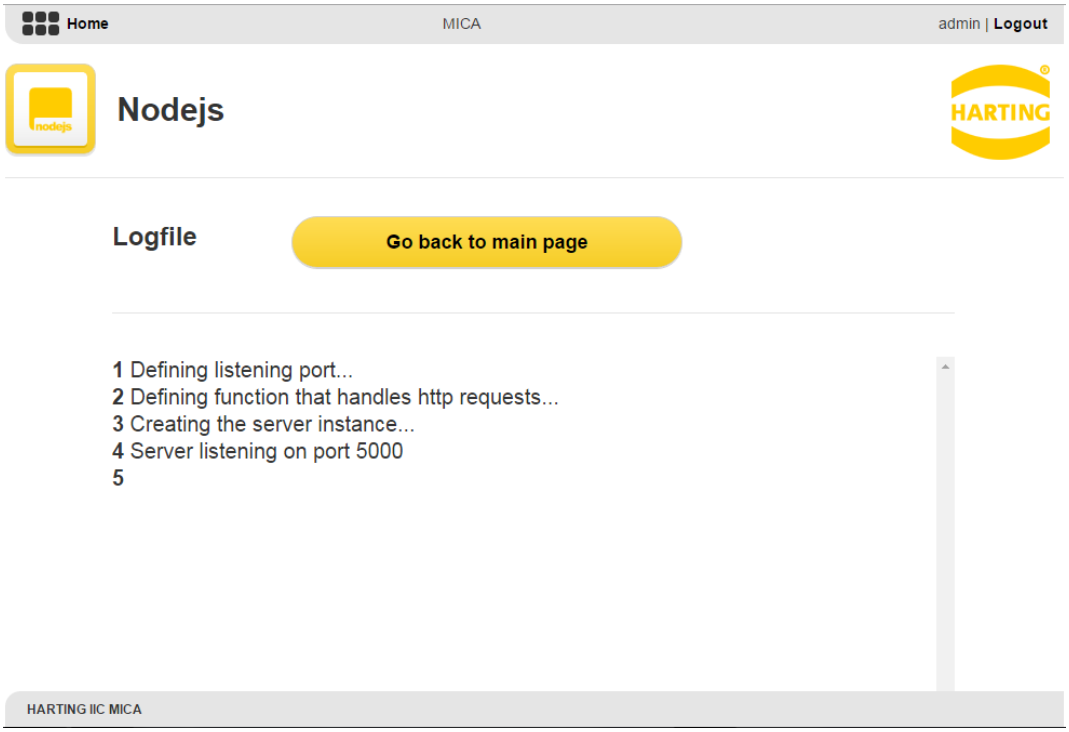

Figure 5: Application logger window of the Node.js Environment Container

3. Usage Example

3.1. Download the default application

The first thing we want to do in this example is to download the default js-application script. Therefore, we simply need to click the “download”-button in the user-interface. When opening the download script file with an editor, you can see that the script implements a simple HTTP web server listening on port 5000. So if you open a browser tab and enter the following address, you should see the response message of the server “It Works!!”.

`http://<container-address>:5000`

The script also writes logs during the initialization process to the file `/var/log/nodeapp.log`. You can see those messages in the logger window when clicking the “Show Log Files” button.

3.2. Modify the application

After we downloaded the application script, we want to download to modify the script by changing the text “It Works!!” to “It still Works!!” and by adding the following line at the end of the script:

```
logger.log('Wow, that was easy!');
```

After that we save the script to "`new_app.js`" and (re-)upload the script as described in the next Section.

3.3. (Re-)upload the modified application

In the last step of our example we will upload the modified application to the container again. Therefore, we click the application file text field, select our script file "`new_app.js`" and finally click the "upload" button. After the file was uploaded we open the logger window again and should see the message "Wow, that was easy!". We then refresh the page of the web server (port 5000) and should see the web server response message "It still Works!!".

4. Services provided by the Node.js container

4.1. Overview

The Node.js Environment Container allows to access its functionality as provided by the user-interface remotely over HTTP requests. The container therefore utilizes the token provided by the IIC MICA host to authenticate the request. For more information see the Section "Single sign-on (SSO)" of the "MICA Programming Guide".

4.2. Usage of the services

The following HTTP-requests are provided by the container:

1. Upload: Uploading a js-application script.
   Method: `POST`
   URL : `appupload?token=XXX`

2. Download: Receiving the content of the current active js-application script.
   Method: `POST`
   URL : `appdownload?token=XXX`

3. Log file: Receiving the content of the application log file.
   Method: `GET`
   URL : `applogfile?token=XXX`

5. Extending the Node.js container

The Node.js Environment Container can be extended by using the provided command line tool `npm`, which gives you access to all open source projects that are available in the npm ecosystem. You can access `npm` as well as the Node.js command `node` after logging into the container via SSH. You then have to navigate to the application folder by typing the following line:

```
cd /opt/nodeapp
```

In this folder you will also find the `app.js` script which is running in the background. To install a new npm-module you simply have to type the following line:

```
npm install <module>
```

Replace `<module>` with the npm-module you would like to install (f.e., `express`).

After installing a new npm-module you can use it within your js-application script by integrating it with the `require(<module>)` command. So if we would like to use express, e.g., we simply need to add the following line to our application:

```
var express = require('express');
```

For more information about npm visit https://docs.npmjs.com/.