|  |  |
| --- | --- |
|  | Francotyp-Postalia |

Windows IoT – Development Environment Setup

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| # | Version | Date | Authors | Comments/Modifications |
|  | 1.0 | 01.09.2016 | Hendryk Irmischer | initial revision |
|  |  |  |  |  |
|  |  |  |  |  |

This document describes the steps to setup a development environment to develop a Node.js application on an IoT-Device (e.g.: Raspberry Pi 3) and how to start debugging.

Table of Contents

[1. Requirements 2](#_Toc460511488)

[1.1. Hardware 2](#_Toc460511489)

[1.2. Software 2](#_Toc460511490)

[2. Basic Setup 2](#_Toc460511491)

[3. Advanced Setup 2](#_Toc460511492)

[3.1. Setup of streamlined development process 3](#_Toc460511493)

[3.2. Debugging 4](#_Toc460511494)

# Requirements

## Hardware

* PC with Windows 10
* IoT Device (supported by Windows 10, e.g.: Raspberry Pi 3)
* FM's, Scales etc

## Software

* Windows 10
* Windows 10 IoT Dashboard
* Node.js
* Visual Studio 2015
  + Node Tools for Visual Studio (NTVS)
  + Command Runner Extension
  + Windows SDK

# Basic Setup

1. See [Windows 10 IoT Docs - IoT Dashboard](https://developer.microsoft.com/en-us/windows/iot/docs/iotdashboard) for instructions on how to install Windows 10 IoT on the Device
2. See [Windows 10 IoT Docs - PowerShell](https://developer.microsoft.com/en-us/windows/iot/docs/powershell) for instructions on how to connect to the Device via PowerShell
3. Clone Solution from:

ssh://git@rdm2.mentana-net.de/fpaas/navigator-web.git

1. Open the solution, right click on the Node project, select Properties and set the "Remote Machine"'s IP-Adress
2. Press F5 (Start Debugging)
   1. VS will no build the project and start a remote debugging session with the device (which can take some minutes)
   2. Remember your deployment folder for the setup of a streamlined debugging process (see doc\Windows IoT - Dev Environment Setup.docx)
3. You can now modify project files and every modified file will be deployed automatically
   1. For static files (e.g.: html, css, images) you don’t even need to restart the node application

# Advanced Setup

Visual Studio 2015 doesn’t reliably recognize which files in the project have changed or not. Even when only modifying some typescript files and leaving all files in node\_modules untouched, VS2015 will often create a new deployment layout. When creating a new layout, VS will delete the whole deployment folder and copy all files again, **which can take several minutes**, depending on the quantity of installed node modules.

***A documentation or something similar, that explains under which circumstances a new layout is being created (which deletes all existing files, and copies them again) or respectively how this behavior can be influenced would be very helpful.***

## Setup of streamlined development process

Step 1: Installation of Chakra Node (Node mit ChakraCore) on the device

* <https://github.com/nodejs/node-chakracore/releases>
* Downloads 🡪 ARM-Packages
* Entpacken z.B.: nach ’C:\Node (Chakra)’

Step 2: Permanently add Node-Folder to PATH

* Read PATH from registry

(Get-ItemProperty -Path ‘Registry::HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment’ -Name PATH).path

* Store old PATH in variable

$oldpath = (Get-ItemProperty -Path ‘Registry::HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment’ -Name PATH).path

* Defined a new variable and append the the node installation folder to it

$newpath = “$oldpath;c:\Node (Chakra)”

* Persist the new value in registry

Set-ItemProperty -Path ‘Registry::HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment’ -Name PATH –Value $newPath

* Double check modified PATH variable

(Get-ItemProperty -Path ‘Registry::HKEY\_LOCAL\_MACHINE\System\CurrentControlSet\Control\Session Manager\Environment’ -Name PATH).path

* Restart PS-Session and check if node can be executed

node –v

* Deploy (Start Debugging) once and remember the deployment folder on the device

e.g.: c:\data\users\DefaultAccount\AppData\Local\Packages\NavigatorWeb\_8f0qdkcs9hxtj\LocalState

* Create network share on the device for easier access to deployed files

net share DeploymentFolder=c:\data\users\DefaultAccount\AppData\Local\Packages\NavigatorWeb\_8f0qdkcs9hxtj\LocalState /GRANT:Everyone`,FULL /REMARKS:”Navigator Web Deployment Folder”

* Map shared folder on the device as network drive on the development machine

e.g.: W:\

* Install the command task runner extension in Visual Studio and open the TaskRunnerExplorer
* Open file commands.json in “[SolutionFolder]\Scripts”
  1. The Deploy-Task defined in here executes a powershell script, that copies all modified project files to the deployment folder on the device via robocopy
  2. To make this work, we need to tell the script/task, where our deployment folder is located via -Dest \"W:/\"
  3. Replace W:/ with the path of your deployment folder (mapped network drive or network path)
* From now on, the project is being deployed automatically after every build

## Debugging

* To debug the application, follow these steps:
  1. Device 🡪 Enter-PSSession
  2. Navigate to “deployment folder”\LocalState
  3. Run: ‘node bin/www’ where “www” is the startup javascript file that bootstraps our application