



# HTML 5 UIs for FDI DEVICE PACKAGES

Application Guide



**FIELDCOMM GROUP™**  
*Connecting the World of  
Process Automation*

**Release Date:** January 28, 2020

**Document Distribution / Maintenance Control / Document Approval**

To obtain information concerning document distribution control, maintenance control, and document approval please contact FieldComm Group at the address shown below.

**Copyright © 2020 FieldComm Group**

This document contains copyrighted material and may not be reproduced in any fashion without the written permission of FieldComm Group.

**Trademark Information**

FieldComm Group™, FOUNDATION™ Fieldbus, HART-IP™, FDI™ are trademarks, and HART®, WirelessHART®, ROM® and SIF® are registered trademarks of FieldComm Group, Austin, Texas, USA. Any use of these terms hereafter in this document, or in any document referenced by this document, implies the trademark/registered trademark. All other trademarks used in this or referenced documents are trademarks of their respective companies. For more information, contact FieldComm Group at the address below.



Attention: FieldComm Group President  
FieldComm Group  
9430 Research Boulevard  
Suite I-120  
Austin, TX 78759, USA  
Tel: +1 (512) 7792-2300

<http://www.fieldcommgroup.org>

**Intellectual Property Rights**

The FieldComm Group (the Group) does not knowingly use or incorporate any information or data into the HART, FOUNDATION Fieldbus and FDI protocol standards, which the Group does not own or have lawful rights to use. Should the Group receive any notification regarding the existence of any conflicting private IPR, the Group will review the disclosure and either (A) determine there is no conflict; (B) resolve the conflict with the IPR owner; or (C) modify the standard to remove the conflicting requirement. In no case does the Group encourage implementers to infringe on any individual's or organization's IPR.

## Contents

Introduction.....	3
User Interface Plugins (UIP) in FDI .....	4
The Road to HTML5 .....	4
The Technologies Behind HTML5 UIPs .....	5
What are the HTML5 technologies?.....	5
Why it is so great? .....	6
How does it work in practice? .....	6
What HTML5 features do UIPs support?.....	7
And what about cyber security? .....	7
Writing an HTML5 UIP .....	8
First Steps .....	8
Creating the HTML5 content .....	9
A note on editing tools .....	9
Creating a UIP Package .....	15
Create a New UIP Project .....	15
UIP Packaging Project Settings .....	16
UIP Variant Settings.....	16
UIP Project Package ID .....	17
Creating an FDI Package for the HTML5 UIP .....	18
Create an FDI Device Package .....	19
Adding the HTML5 UIP to a Device Package .....	20
Adding a Plug-In to your EDDL Source .....	21
Build the FDI Device Package .....	22
Testing the HTML5 UIP .....	23
How can you debug the UIP? .....	23
Implementing HTML5 UIPs on FDI Hosts .....	24
Support and Resources offered by FieldComm Group .....	26



## Introduction

Field Device Integration (FDI) is the technology which allows for a seamless and effortless integration of data from smart field devices into your DCS system, maintenance and optimization systems or the cloud. Device vendors provide one FDI Device Package per protocol for one device. These FDI Device Packages can be imported by all FDI hosts, allowing users to operate the device. An FDI host can be a standalone software component on a workstation PC or a mobile device, an integrated software component in a distributed control system or a software component with a client server architecture. FDI host examples include dedicated products such as handheld field communicators, standard software components of a distributed control system like device management software or optional software systems such as plant asset management systems, historians, and analytics packages.

An FDI host typically consists of an FDI client, an FDI server and one or more FDI communication servers.

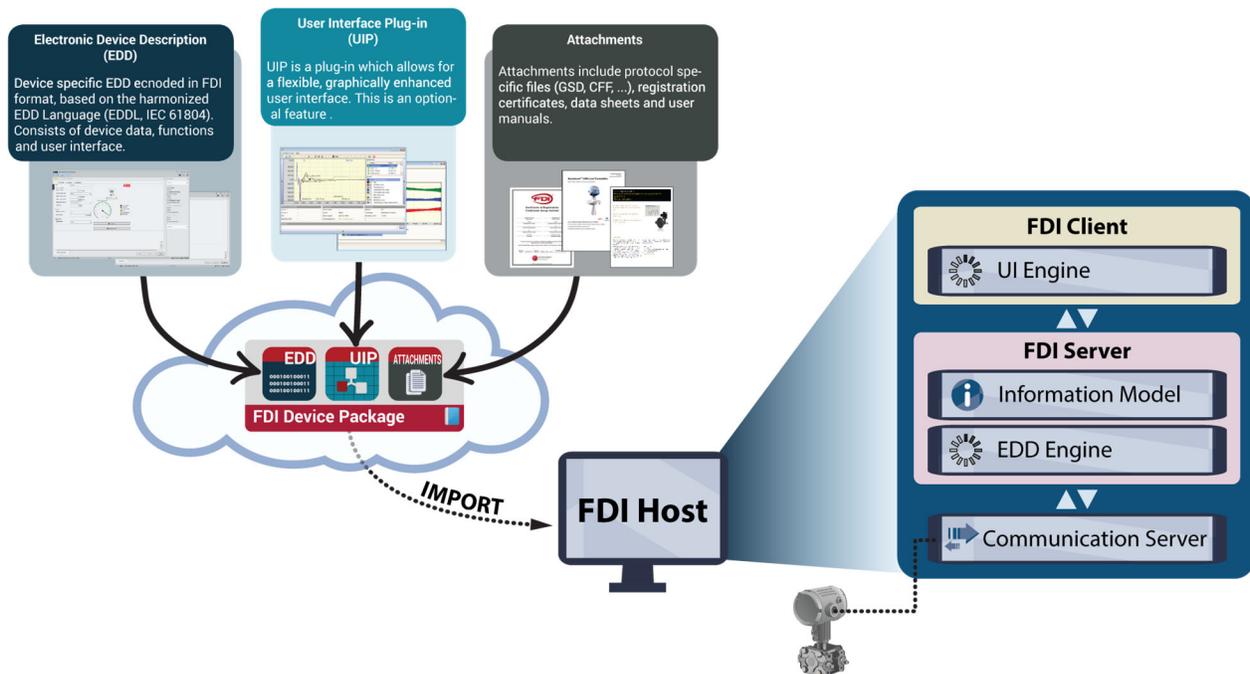


Figure 1. FDI Host

## User Interface Plugins (UIP) in FDI

In addition to the Electronic Device Description (EDD), which is part of any FDI Device Package and allows for a uniform description of the device including user interfaces for device configuration and diagnostics, any FDI Device Package may include one or several User Interface Plugins. Compared to the EDD, UIPs may provide the following extended functionalities:

- flexible, graphically enhanced user interfaces
- custom logic, e.g. for diagnostics
- direct access to the device, e.g. for a firmware update
- import, export and display of files, e.g. to generate, save and show reports

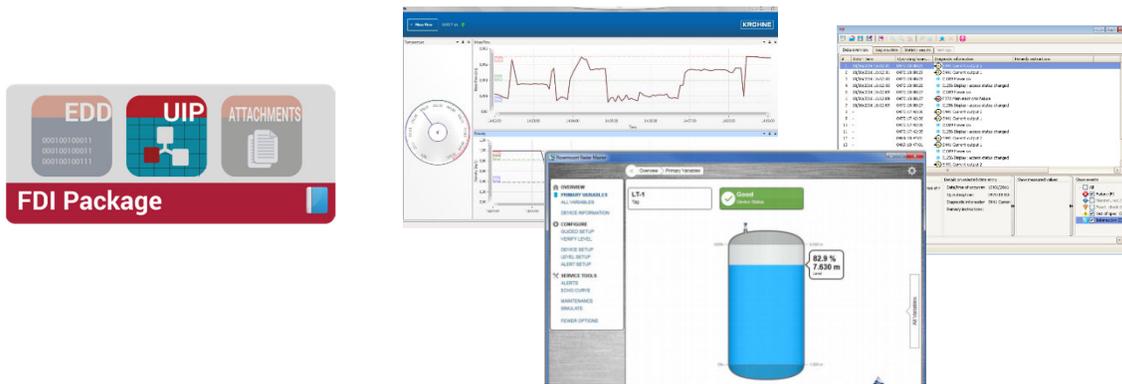


Figure 2. User Interface Plugins (UIP)

Since FDI hosts may run in various environments, FDI incorporates a technology mapping and platform concept to ensure UIPs can be executed on the host systems without interoperability issues or compromising the system stability. Based on the FDI Platform concept host vendors can target the User Interface Plugins to specific use cases and environments. With version 1.2 of the FDI Technology Standard, FDI specifies the following two technology mappings to provide the means to cover various host environments: .NET and HTML5. Device vendors can choose the technology that is best suited to their requirements and target platforms. This whitepaper provides the information to make the choice.

For both UIP technologies, FDI specifies the interface between the UIP and the FDI Client, which will host and execute the UIP. The services available within this interface are identical for both technologies. Using these services, UIPs may access data of the device and provide the extended UIP functionalities like open or store data on the FDI Client. This whitepaper will give you a first impression of how to implement a UIP with such functionalities.

## The Road to HTML5

In the beginning, FDI was mainly about bringing existing Technologies like EDDL and FDT together to build a more harmonized approach to the major communication technologies for the process industries (HART, FOUNDATION Fieldbus, Profibus, Profinet, Modbus, etc.). With HTML5, UIPs FDI is opening to a much more user-centric approach. The HTML5 technology makes it possible to use different end user devices like smartphones, tablets, laptops or big screens without the need of creating individual package variants.

Basically, everything which has been possible with .NET based UIPs can be done with HTML5 UIPs as part of an FDI package. And everything will run on a well proven web engine which is part of every new FDI host. For example, the following picture shows a complex device configuration view running completely in HTML5:

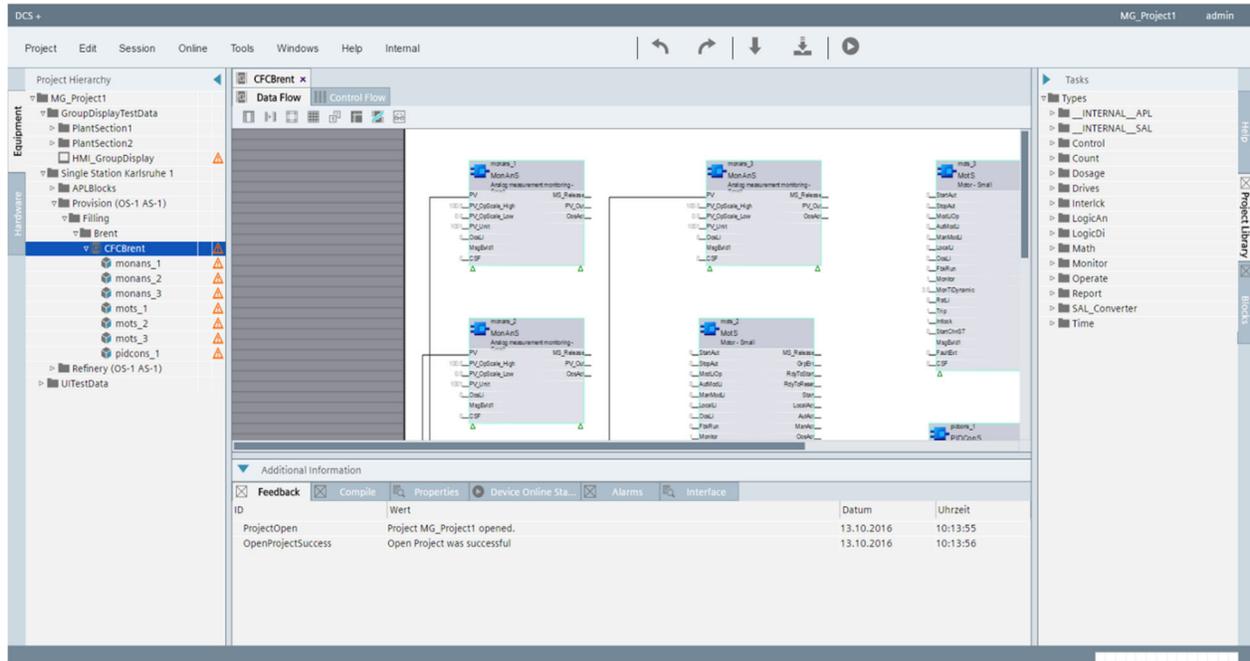


Figure 3. Configuration of a device in an HTML5 view

## The Technologies Behind HTML5 UIPs

With FDI Technology Version 1.2, FDI supports User Interface Plugins (UIPs) written in HTML5 technologies via the [FDI Integrated Developer Environment](#) (release 1.4.0 or later).

### What are the HTML5 technologies?

HTML5 technology is not a single technology, but rather the name for a set of current state-of-the-art technologies for web applications:

	<p>HTML5: The latest evolution of the standard that defines HTML. Its main features are Semantics, Connectivity, Offline and storage, Multimedia, 2D/3D Graphics and effects, Performance and Integration, Rendering device access, and Styling.</p>
	<p>CSS3: This is the latest evolution of the Cascading Style Sheet for styling web pages. Version 3 provides long-awaited novelties, like rounded corners, shadows, gradients, transitions and animations, as well as new layouts like multi-columns, flexible box or grid layouts.</p>
	<p>JavaScript (JS): This is a lightweight, interpreted or JIT (just-in-time) compiled programming language supported by browsers for the usage in web pages. JS is a prototype-based, multi-paradigm, dynamic scripting language, supporting object-oriented, imperative, and declarative programming styles. The standard for JavaScript is ECMAScript.</p>

	<p>TypeScript: TypeScript is an open-source programming language developed and maintained by Microsoft. It is a superset of JavaScript and adds additional static typing to the language. During development typescript code gets compiled (transpiled) into JavaScript. TypeScript makes development of web applications easier and more effective, by offering better structuring of the code and type safety.</p>
	<p>JSON: JavaScript Object Notation (JSON) is an open-standard file format that uses human-readable text to transmit data objects of attribute-value pairs and array data types. It is a very common format for storing and transporting data.</p>
	<p>Content Security Policy (CSP) is an added, standardized layer of security used by state-of-the-art web applications as, for example, home banking applications.</p>

### Why it is so great?

There are two main advantages of web user interfaces when compared to legacy toolkits like .NET, Java or C++-based user interfaces:

- Web UIs are not bound to a specific platform or runtime. One UI can run on different platforms (Linux, Windows, Android, IOS) and on any device – only a browser engine that supports HTML5 is required.
- With the HTML5 layout features, one can design user interfaces that automatically adapt to different screen resolutions and ratios as shown in the following images.



Figure 4. HTML5 page with 1-column layout

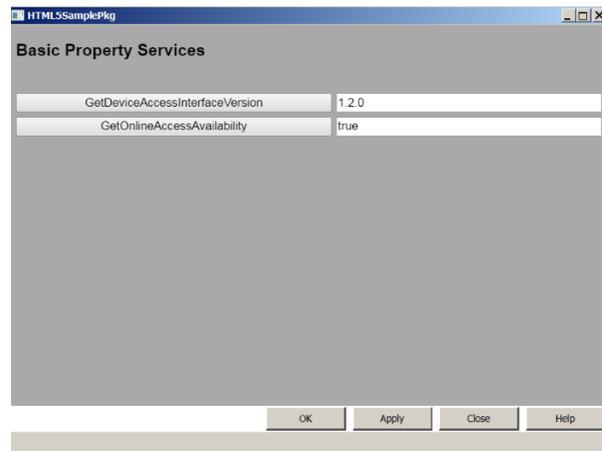


Figure 5. HTML5 page with 2-column layout (same code)

This makes it possible to run the same UIP on a mobile device, tablet, laptop or workstation, or a big screen in an operations center.

### How does it work in practice?

An HTML5 UIP runs like any other User Interface Plugin from a menu entry that is part of the device EDD and is rendered by the FDI host.

When the menu item is triggered, the FDI host starts a UIP host frame application containing an embedded HTML5 browser engine. In the HTML5 browser engine the HTML5 UIP is loaded:

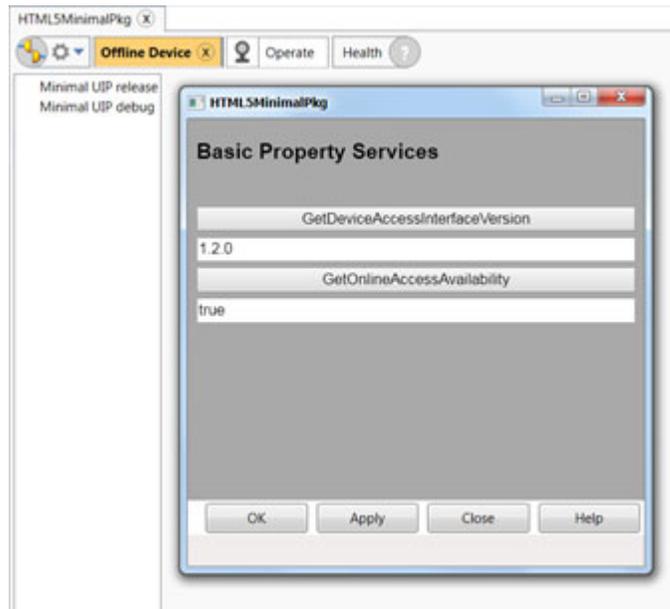


Figure 6. HTML Browser with HTML5 UIP Loaded

When some information is needed from the device the HTML5 UIP uses the FDI type library, which is defined in the FDI Standard, to get the information. With FDI version 1.2, the FDI type library is defined in TypeScript and JavaScript making it possible to use the FDI type library with HTML5 pages and JavaScript code.

### What HTML5 features do UIPs support?

Since HTML5 technologies evolve quickly and browser engine suppliers don't always support the same HTML5 features, it is important to agree on a minimum set of HTML5 features which are guaranteed to work on every FDI Host. This minimum feature set has been defined by the FDI Standardization team and is available in the [FDI Specifications](#).

All defined HTML5 UIP features are supported by the major browser engines (Chromium, Firefox, Webkit). This means that an HTML5 UIP written with the minimum feature list is guaranteed to run on every FDI host. On the other hand, it is not prohibited to use an HTML5 feature that is not listed in the minimum feature set. However, in this case the FDI package supplier needs to make sure that additional checks and workarounds are in place if a browser engine or the FDI host does not support the feature (e.g. by using polyfills). For instance, if a UIP offers barcode scans from a camera, the UIP should check first whether the FDI host supports taking pictures since cameras may not be available on every FDI host hardware, such as a PC. If no camera is present, a work around would be entering the barcode number with the keyboard.

### And what about cyber security?

Security in an HTML5 UIP is addressed in various ways:

- The embedded web browser engine provides a sandbox environment for running the HTML5 UIP. Direct communication with other applications or the operating system is restricted, checked or prohibited – access to external resources (camera, file system, geolocation) is only possible

through special JavaScript APIs such as Geolocation API, Orientation API or Web Audio API.

- To protect the HTML5 UIP from internet attacks, the HTML5 UIP host supports Content Security Policy (CSP) – a computer security standard introduced to prevent cross-site scripting (XSS), clickjacking and other code injection attacks resulting from execution of malicious content in the trusted web page context. Note that CSP does not allow inline script or inline CSS stylesheets in an HTML file. This also leads to a clearer architecture of the web application.
- FDI packages are protected from tampering by digital signatures. An FDI package contains two signatures:
  - One signature is generated by the FDI package originator and is intended to provide assurance that the package content is authentic and has not been modified by an unauthorized entity.
  - The other signature is created by the FDI Package Registration Authority (FieldComm Group or Profibus International (PNO)) and is intended to provide assurance that the relevant parts of the package (i.e. the UIP(s) and the EDD) have passed the conformity tests. Since a UIP is part of an FDI package, the protection against tampering is extended to UIPs as well.
- The JavaScript FDI type library used for the communication between the HTML5 UIP and the UIP host is not changeable since it is part of the assemblies of the UIP host and these assemblies are signed.

## Writing an HTML5 UIP

So how can you do all this?

This section provides step by step guidance for creating your first HTML5 UIP integrated into an FDI package. You must have an FDI Package IDE installed on your PC or laptop. It can be purchased at [fieldcommgroup.org](http://fieldcommgroup.org).

### First Steps

First, you should specify the structure and design of your UIP user interface as you would for creating your new company website. Bear in mind what devices you want your UIPs to be used with and how to implement an attractive responsive design for all of these devices.

Next, you should look at the sample HTML5 UIP which is installed with the FDI Package IDE under MyDocuments/FDI/IDE/Examples/UIP (see screenshot below). HTML5 UIP sample not only contains the UIP itself, but also additional files which are helpful in the creation of HTML5 UIPs.

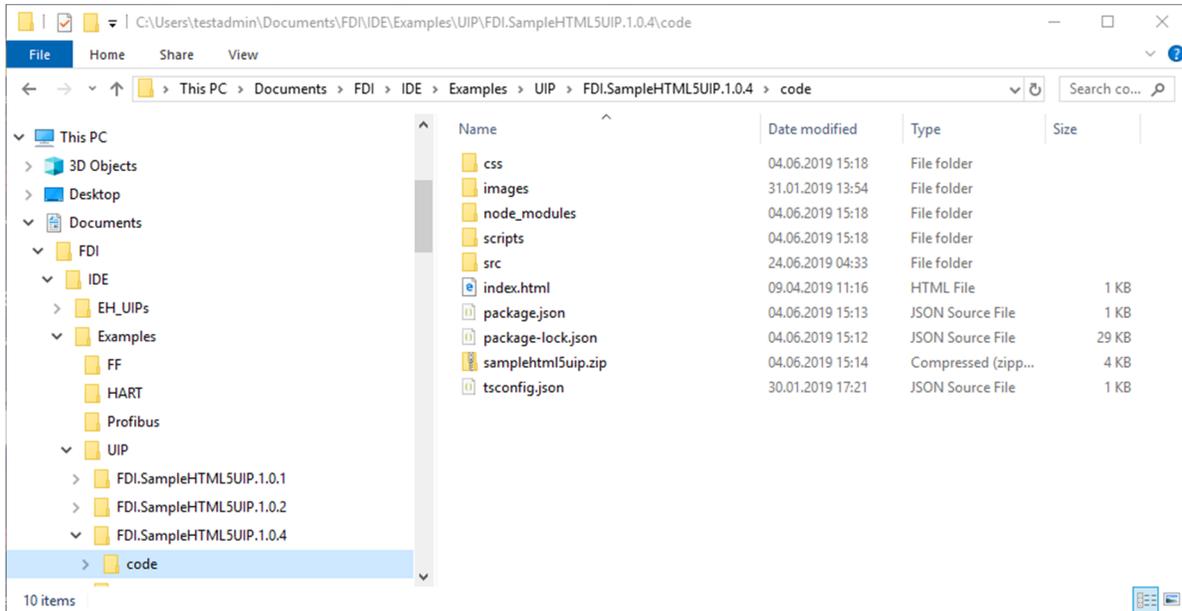


Figure 7. Location and structure of the Sample HTML5 UIP

The best course of action here is to make a copy of the sample HTML5 UIP and use it for creating your own HTML5 application content.

The provided Sample UIP contains HTML5 source files for the UIP, the FDI HTML5 UIP Type Library, a CSS file, and a TypeScript compiler. Open the code folder in Visual Studio Code to begin.

## Creating the HTML5 content

### A note on editing tools

*For editing the content you can use any text editor. Developers can choose generic text editors like Notepad++ or TextPad, or more advanced ones like Brackets or Visual Studio Code. Advanced tools like content management systems can also be used to create the HTML source.*

*In our example we used Visual Studio Code to edit the HTML, CSS and TypeScript files. Using Visual Studio Code has the advantage that follow up steps (e.g. transpilation and packaging) are already integrated in the IDE.*

As a first step you need to define an entry page HTML file (in our case index.html) for all of your content (HTML code, JavaScript, etc.).

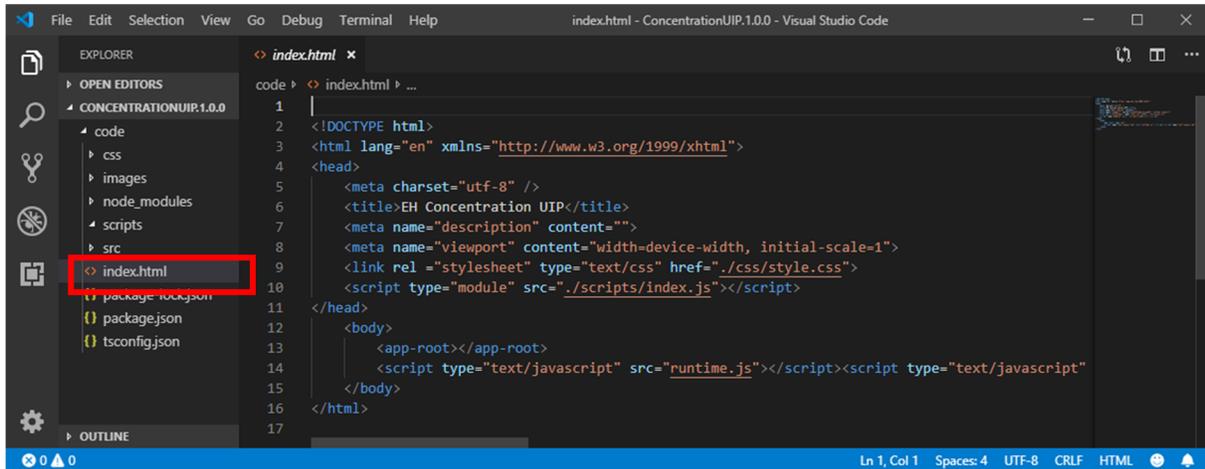


Figure 8. Entry page for the HTML5 UIP (index.html)

The style for your HTML5 pages should always be specified in Cascading Style Sheet (CSS) files. These files can be edited with the same tools.

Put your CSS files together with all other needed files, such as images, into the respective folders of your project folder.

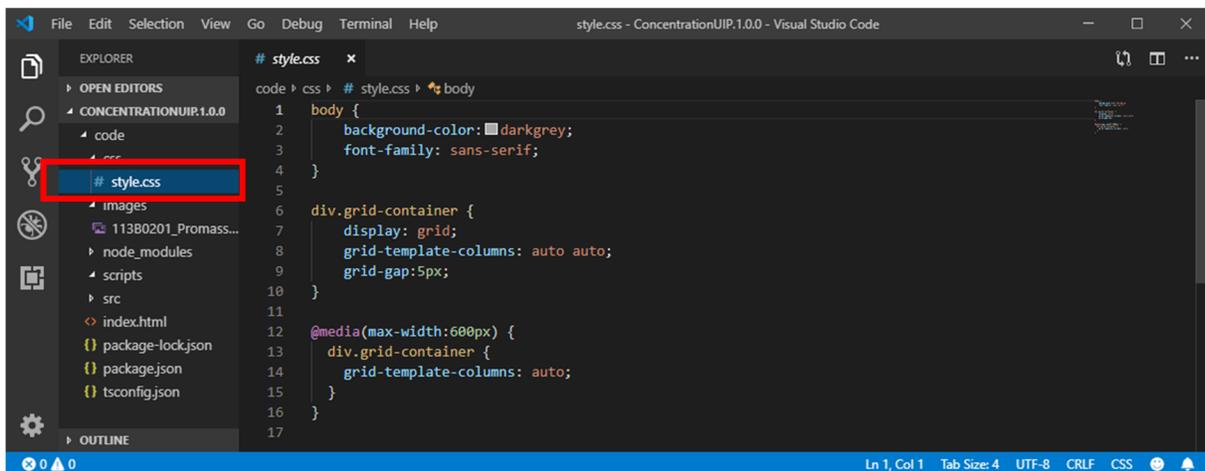


Figure 9. CSS file of the Sample HTML5 UIP

Next, you should have a look at the typescript files in the project. They are contained in the `src` folder of the project.

The `fdi.ts` TypeScript file specifies the FDI type library interface to your FDI host. This and all the other typescript files of the `src` folder except for `index.ts` are provided by the FDI host. You should not change these files because any change will be overwritten by the FDI host. For adding your own code, edit the `index.ts` file. This is the starting point for custom logic in the HTML5 UIP. Within your UIP TypeScript file (`index.ts`) you can specify all function calls to the FDI host using the FDI type library.

In the sample project two simple calls to the FDI host for reading the interface version and the online access availability are defined.

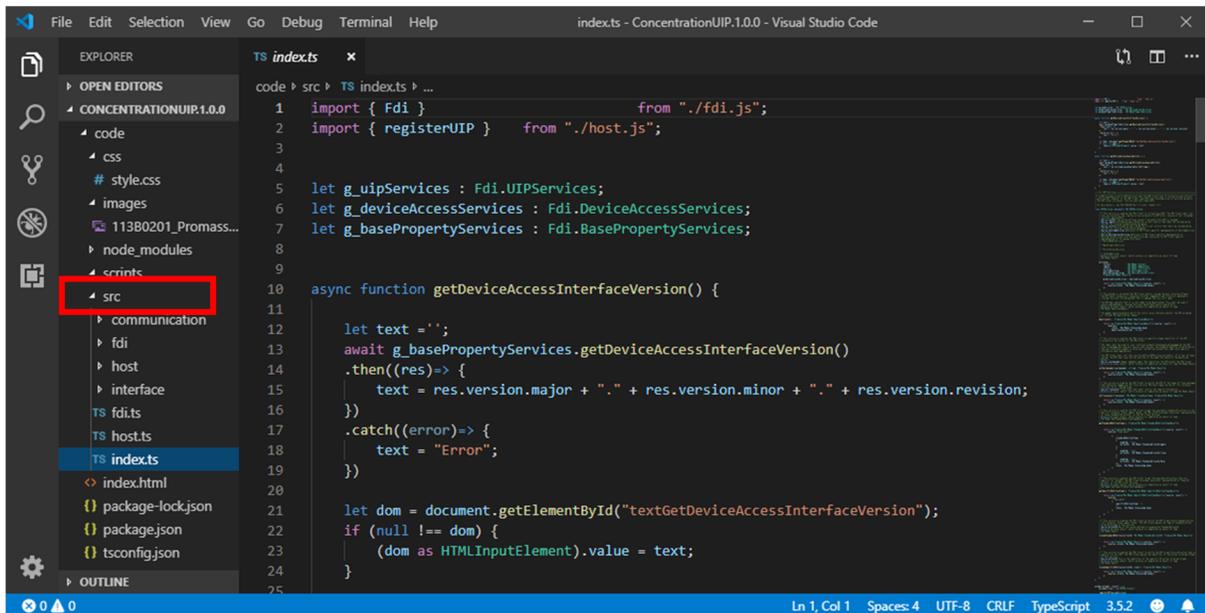


Figure 10. TypeScript src folder

Now it is time to generate the JavaScript files from the TypeScript files. This step is called *Transpilation* or source-to-source compiling to translate the TypeScript to JavaScript.

But before doing the transpilation you should look at the *tsconfig.json* file at the project root. The *tsconfig.json* specifies how all of your TypeScript files are transpiled in JavaScript. For example, *target* defines the JavaScript version of the code that is to be generated, *outDir* specifies the output directory for the JavaScript code.

For more information consult the TypeScript compile options page (<https://www.typescriptlang.org/docs/handbook/compiler-options.html>)

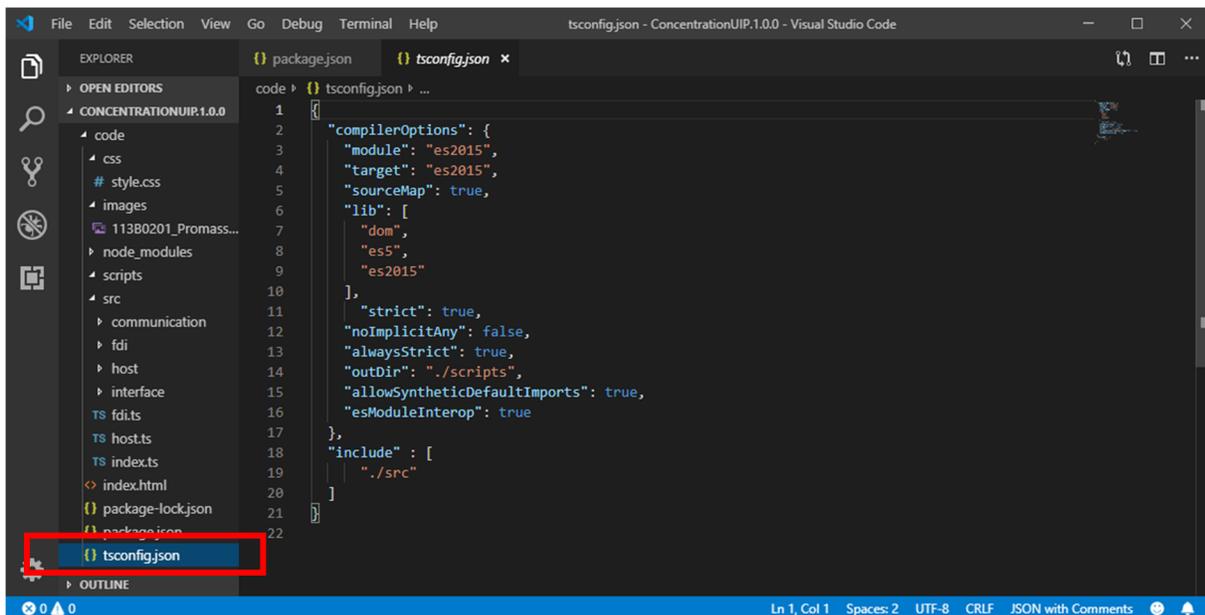


Figure 11. Content of tsconfig.json file

We are ready for the build now.

Use the `tsc:build` task to transpile the TypeScript into JavaScript as shown in the next two screenshots.

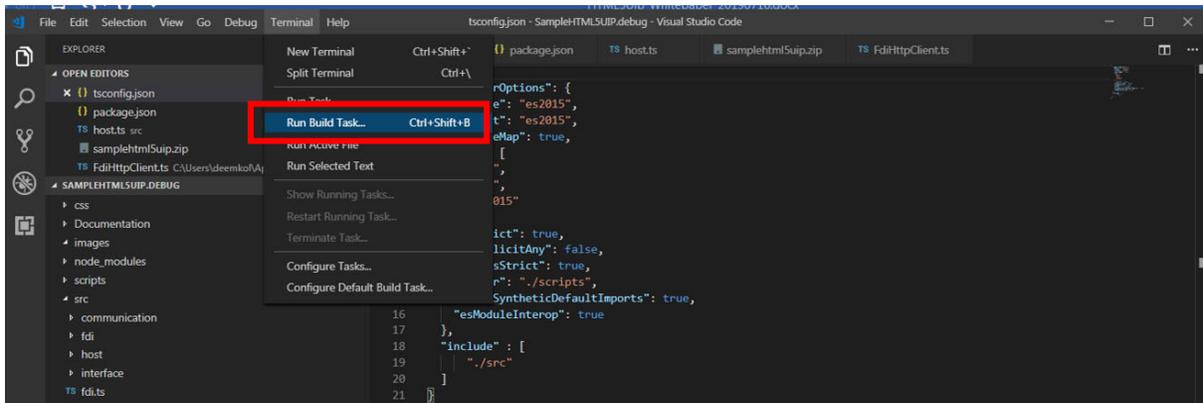


Figure 12. Navigating to the build tasks

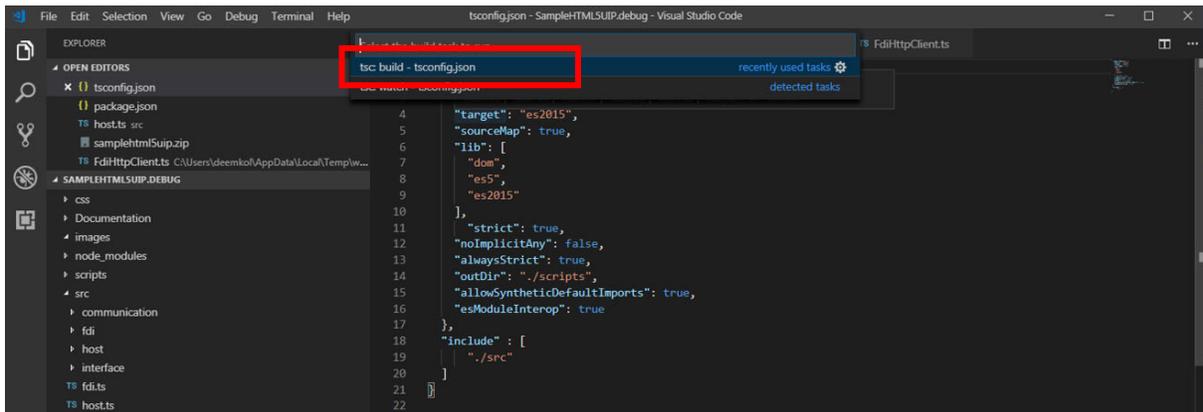
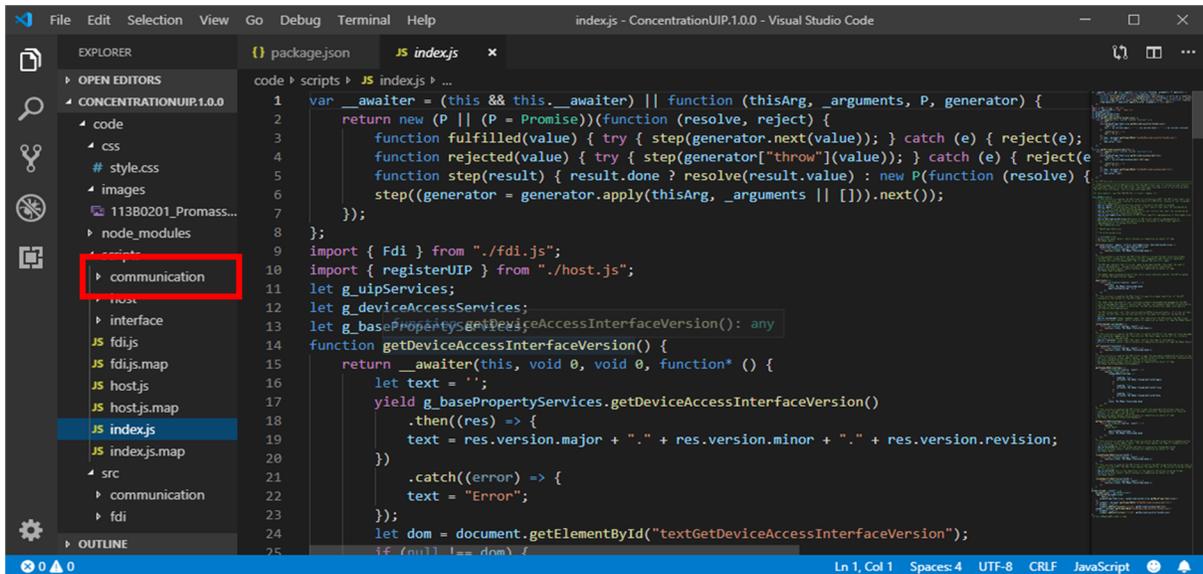


Figure 13. Run `tsc:build` to generate the JavaScript code

After transpilation of the TypeScript files to JavaScript you will find the generated JavaScript files in the "scripts" folder (or wherever you specified in the `tsconfig.json`) as shown below:

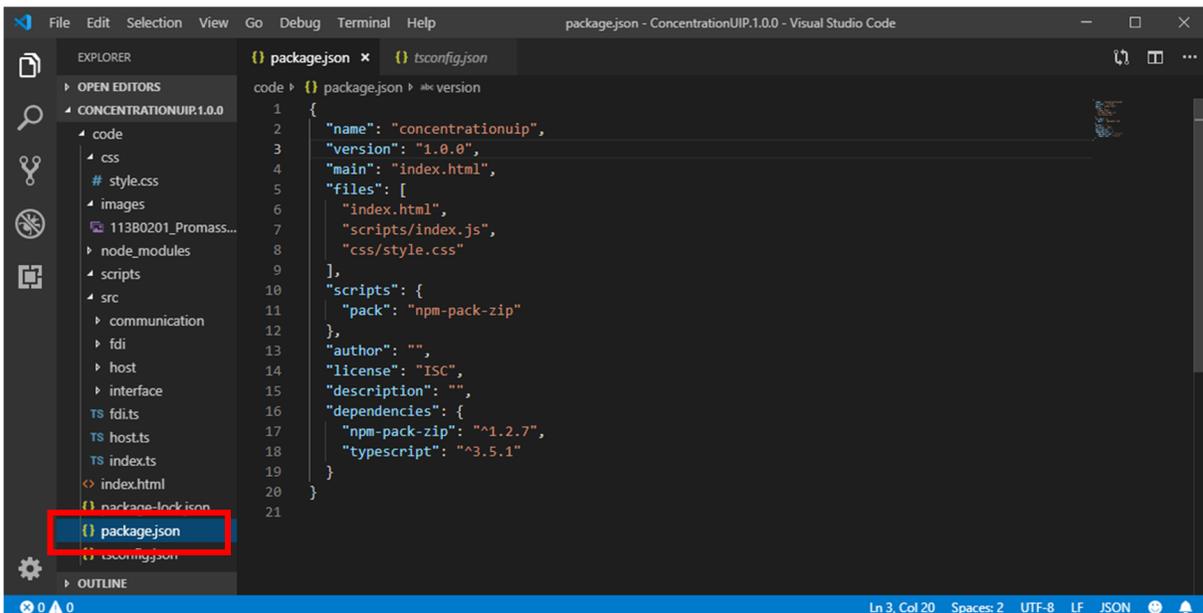
## HTML 5 UIPS FOR FDI DEVICE PACKAGES



```
code > scripts > JS indexjs > ...
1  var __awaiter = (this && this._awaiter) || function (thisArg, _arguments, P, generator) {
2  return new (P || (P = Promise))(function (resolve, reject) {
3      function fulfilled(value) { try { step(generator.next(value)); } catch (e) { reject(e); }
4      function rejected(value) { try { step(generator["throw"](value)); } catch (e) { reject(e); }
5      function step(result) { result.done ? resolve(result.value) : new P(function (resolve) {
6          step((generator = generator.apply(thisArg, _arguments || [])).next());
7      });
8  });
9  import { Fdi } from "./fdi.js";
10 import { registerUIP } from "./host.js";
11 let g_uipServices;
12 let g_deviceAccessServices;
13 let g_basePropertyServices;
14 function getDeviceAccessInterfaceVersion() {
15     return __awaiter(this, void 0, void 0, function* () {
16         let text = '';
17         yield g_basePropertyServices.getDeviceAccessInterfaceVersion()
18             .then((res) => {
19                 text = res.version.major + "." + res.version.minor + "." + res.version.revision;
20             })
21             .catch((error) => {
22                 text = "Error";
23             });
24         let dom = document.getElementById("textGetDeviceAccessInterfaceVersion");
25         if (dom) {
26             dom.textContent = text;
27         }
28     });
29 }
```

Figure 14. JavaScript Source Output

The next step is to package the HTML, CSS and generated JavaScript code into a zip file. Here, the package.json file specifies how all necessary files are zipped together into a zip archive.



```
code > package.json > version
1  {
2    "name": "concentrationuip",
3    "version": "1.0.0",
4    "main": "index.html",
5    "files": [
6      "index.html",
7      "scripts/index.js",
8      "css/style.css"
9    ],
10   "scripts": {
11     "pack": "npm-pack-zip"
12   },
13   "author": "",
14   "license": "ISC",
15   "description": "",
16   "dependencies": {
17     "npm-pack-zip": "^1.2.7",
18     "typescript": "^3.5.1"
19   }
20 }
21 }
```

Figure 15. Content of package.json file

For creating the zip file, just execute the `npm:pack` task in Visual Studio Code

## HTML 5 UIPS FOR FDI DEVICE PACKAGES

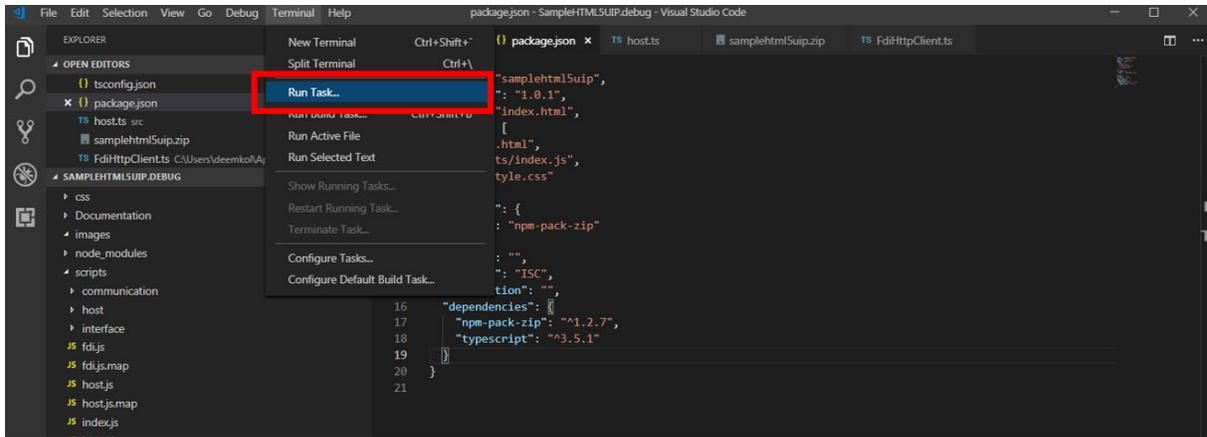


Figure 16. Navigating to the `tsc:pack` task

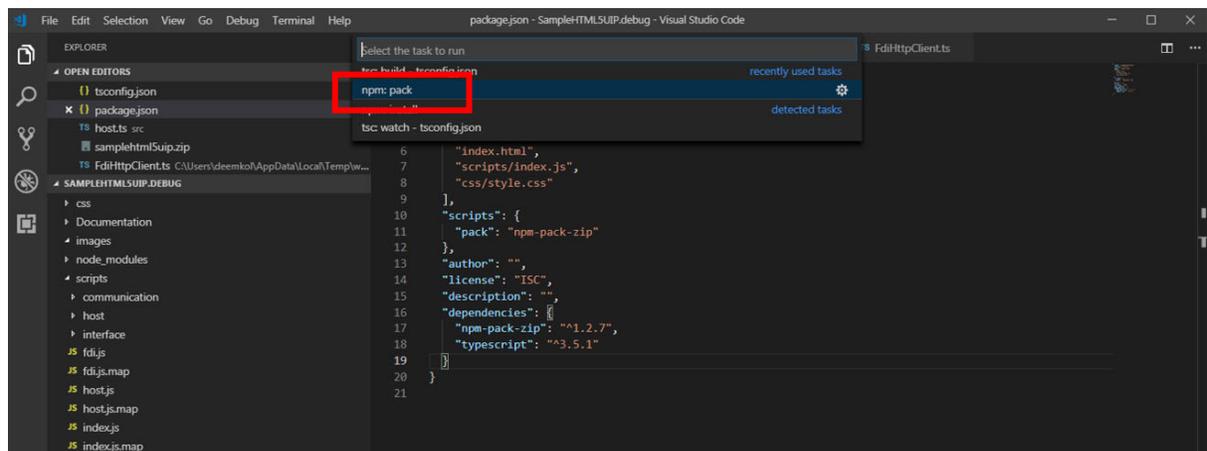


Figure 17. Executing `npm:pack` to create a zip archive

The zip archive, `samplehtml5uip.zip`, is created in the project directory.

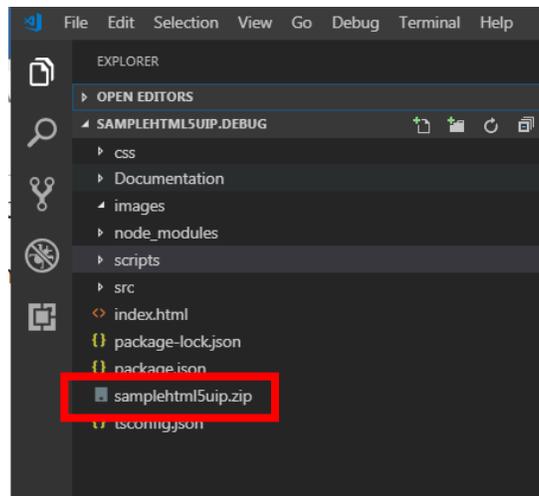


Figure 18. Generated zip file `samplehtml5uip.zip`

You can verify the contents of your zip archive with any standard zip archiver (Win ZIP, 7Zip, etc.):

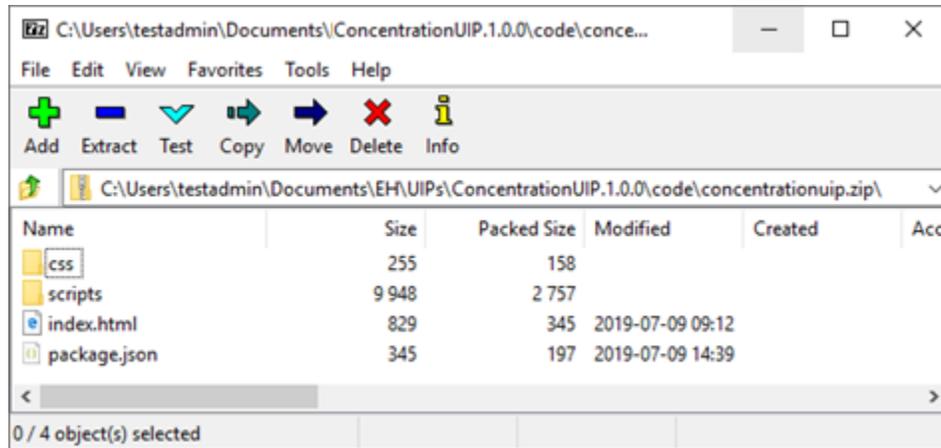


Figure 19. UIP zip archive contents

Note that the UIP project zip file does not contain the *FDI type library* or *FDI host JavaScript files*. These files will be automatically included when the UIP is loaded in the FDI host.

### Creating a UIP Package

Next, you must create your UIP Packaging Project in the FDI IDE. This is the same procedure as for .NET UIPs and described in the User Manual for the FDI IDE.

#### Create a New UIP Project

First, open the FDI IDE and create a New UIP Packaging Project as shown:

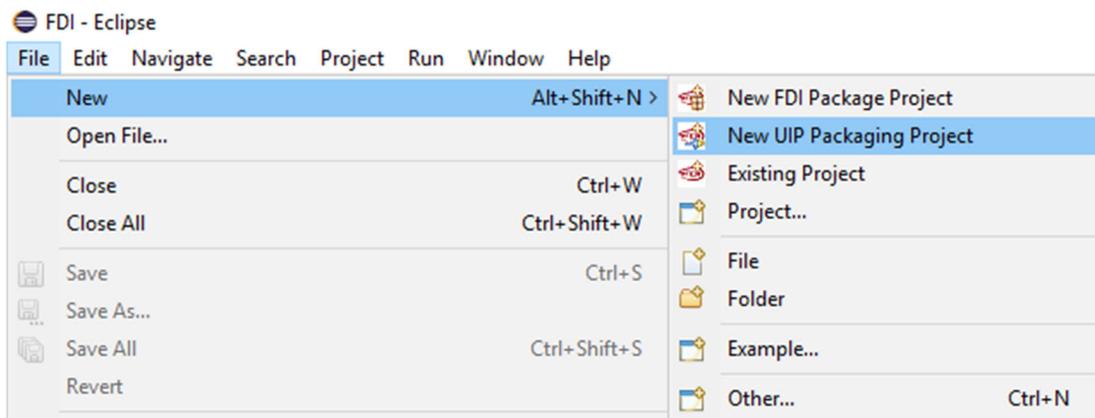


Figure 20. Creation of UIP Packaging Project

Next, add the necessary project details.

## UIP Packaging Project Settings

Note that the *Supported FDI version* must be 1.2.0 as shown in the following image:

The screenshot shows a dialog box titled "New UIP Packaging Project". It contains several input fields and a dropdown menu:

- UIP Project name:** ConcentrationUIP
- UIP Project directory:** C:\Users\testadmin\Documents\EH\UIPs\Concentration (with a "Browse..." button)
- UIP name:** ConcentrationUIP
- Supported FDI version:** 1.2.0 (highlighted with a red box)
- Additional:** (empty text field)
- Style:** DIALOG (dropdown menu)

At the bottom of the dialog, there are four buttons: "?", "< Back", "Next >", "Finish", and "Cancel".

Figure 21. UIP project details

## UIP Variant Settings

In the UIP Variant page of the project settings, you select the zip archive containing your UIP files, remember this zip archive contains the "index.html" file.

For the target Platform you can select if the UIP should run only on Workstations, only on Mobile devices or on both systems. The example shown on the next page has both types of systems selected.

*Start element name* should contain the entry page of the HTML5 UIP (index.html).

**New UIP Packaging Project**

UIP Variant  
Please enter an User Interface Plug-in variant.

Variant: C:\Users\testadmin\Documents\EH\UIPs\Concentratio

Version: 1.0.0

Platform: WorkstationAndMobile

Runtime Id: FdiHtml1

Execution environment: any

Start element name: index.html

List of documents

Figure 22. Details for UIP Variant

Click “Finish” to save the UIP Packaging Project.

### UIP Project Package ID

After creating the UIP Packaging Project you can see the Package ID of the UIP was automatically generated. This Package ID will be used in the next step.

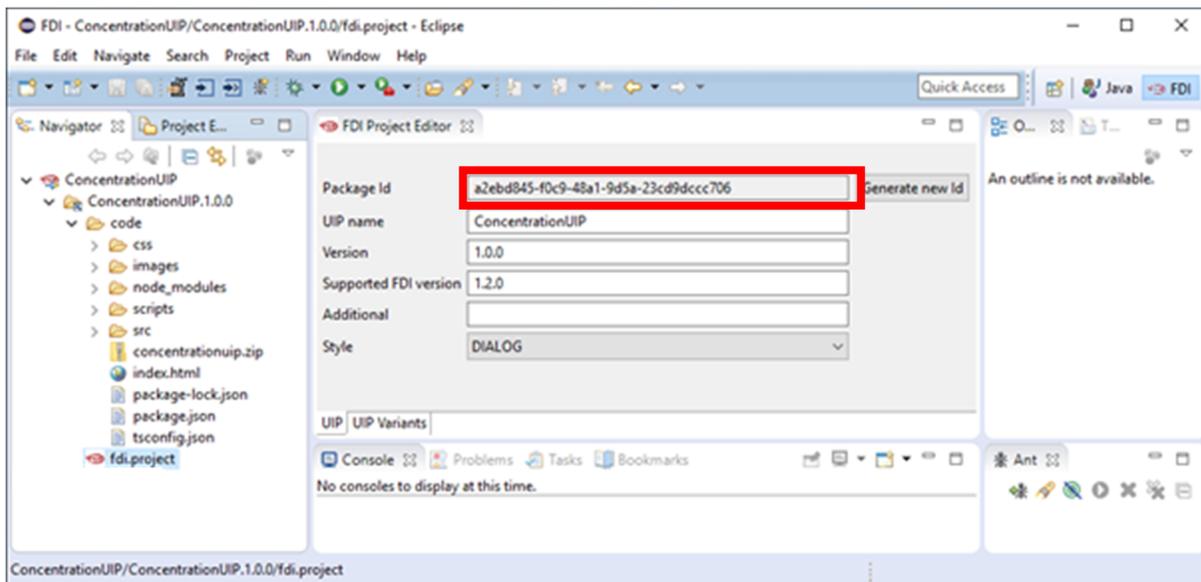


Figure 23. Package ID for the UIP Package

Creating an FDI Package for the HTML5 UIP

Next, the HTML5 UIP will be included in a UIP Package.

Go to the “Project” menu of the FDI IDE. Use the menu item “Create Package” to create the actual UIP Plug-in file (e.g. ConcentrationUIP.1.0.0.uip).

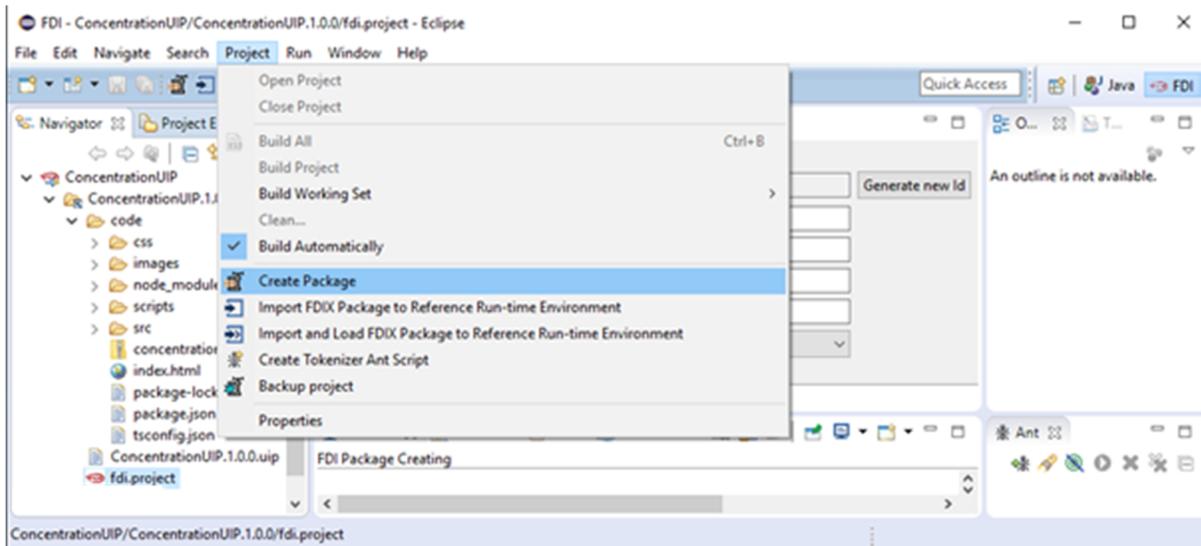


Figure 24. Creation of an FDI Package for a UIP

This UIP Plug-in file can now be referenced either directly from an FDI Device Package or from an FDI UIP Package (which itself could be referenced by several FDI Device Packages).

## Create an FDI Device Package

Using the same FDI IDE tool, create a New FDI Package Project.  
Specify the Package Type (use *Device* this time).

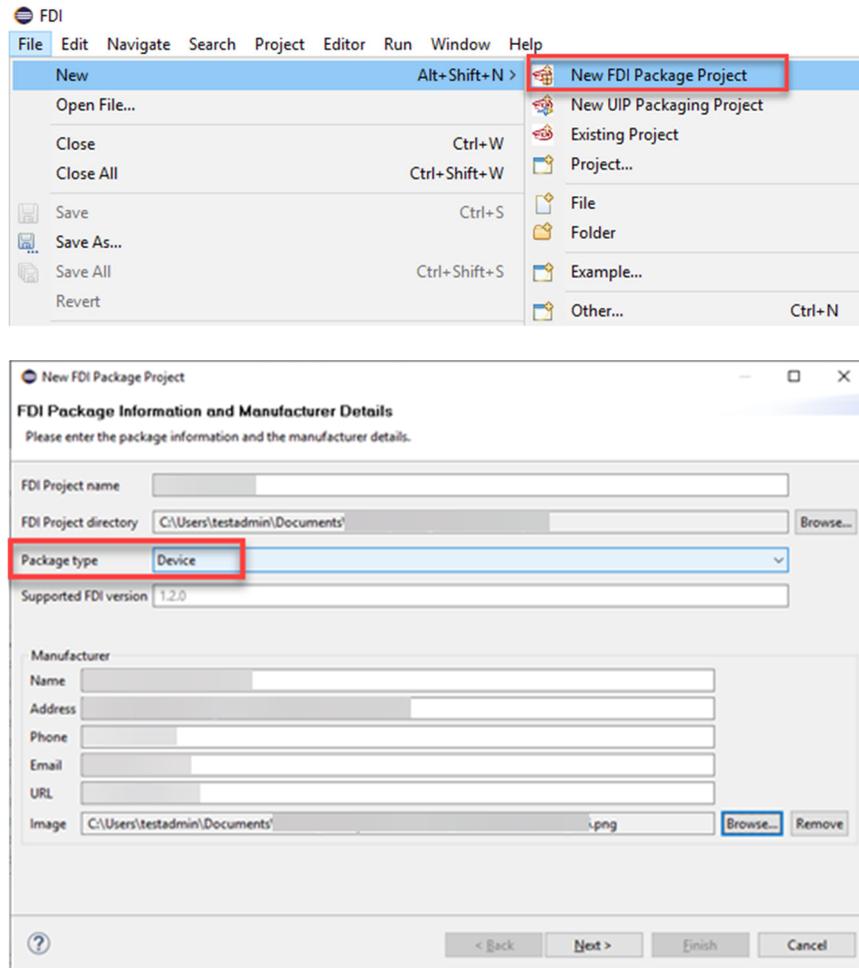


Figure 25. FDI Package Project details – Device Package

In the case of an FDI Device Package, specify the Device Type details as you would do for any normal FDI Device Package:

- EDDL source file (\*.ddl)
- Device Name
- Classification
- Interface
- Protocol Version
- Communication Role (Client)
- Manufacturer ID and Device Type codes

Figure 26. Device type details for an FDI Device Package

### Adding the HTML5 UIP to a Device Package

Within your FDI package project settings you can now add your HTML5 UIP.

Go to the section titled “List of supported UIPs” in the FDI Project Editor for the FDI Device Package you are currently creating. Click on “Add UIP...” and browse for the UIP file (e.g. ConcentrationUIP.1.0.0.uip)

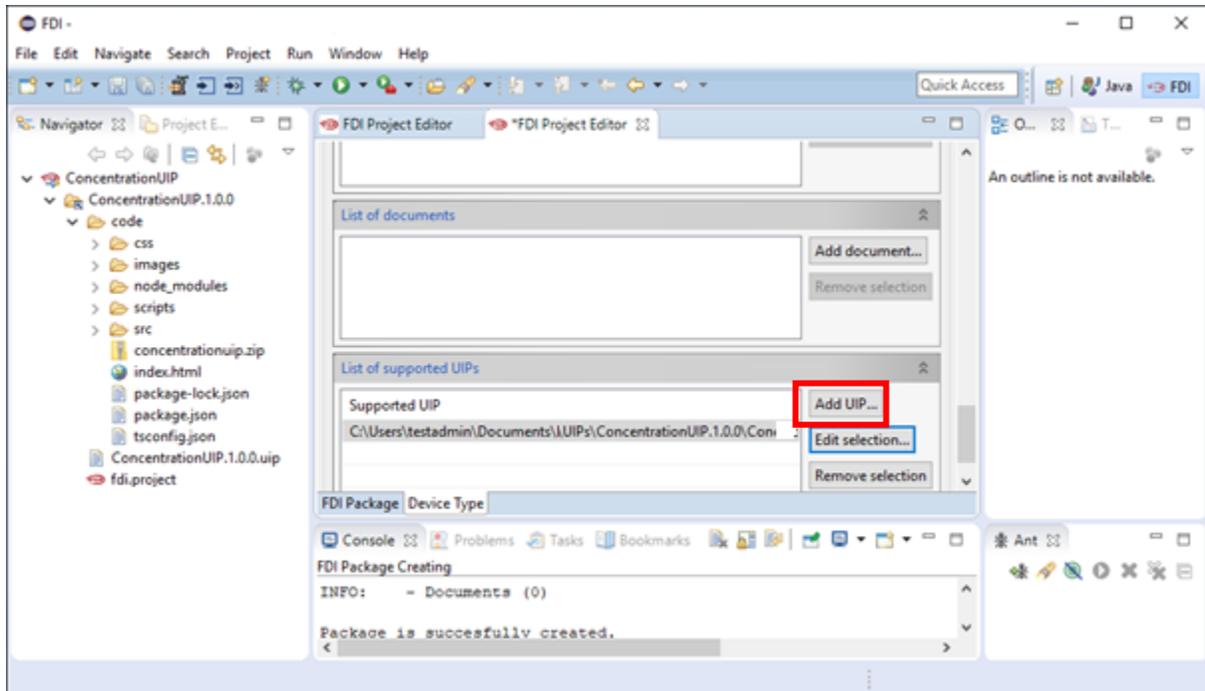


Figure 27. Adding a UIP to the FDI Package

### Adding a Plug-In to your EDDL Source

To reference your HTML5 UIP you need its Package ID from the previous step:

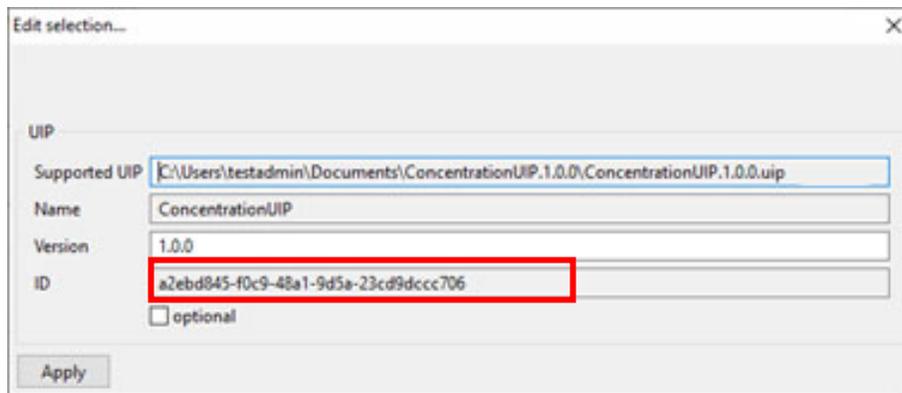


Figure 28. Paste the UIP Package ID

In the EDD source code for the FDI Device Package, add a PLUGIN to refer to your HTML5 UIP.

Using the UIP Package ID as the “UUID” of the PLUGIN, you can edit the device EDD code to add a trigger for the UIP in the offline root menu, for example:

```

PLUGIN ConcentrationUIP
{
    LABEL "ConcentrationUIP";
    UUID a2ebd845-f0c9-48a1-9d5a-23cd9dccc706; //enter the UIP's UUID here (ID from FDI Project editor tab 'UIP')
}

MENU offline_root_menu
{
    LABEL MENU_OFFLINE_ROOT_DEVICE;
    STYLE MENU;
    ITEMS
    {
        ConcentrationUIP
    }
}

```

Figure 29. Menu entry and trigger for starting the HTML5 UIP

The UIP Package must appear on a visible menu for the user to access it.

### Build the FDI Device Package

Save all edits made to the EDDL source code. Save all changes to the FDI Project settings (e.g. fdi.project). Saving the project will automatically regenerate the Tokenizer Ant Scripts. You can also manually update the Ant Scripts by choosing the “Project” menu and then “Create Tokenizer Ant Script”.

Run the Ant Scripts to Tokenize the EDDL source and to build the FDI Device Package. (See the FDI IDE User Guide for detailed instructions.)

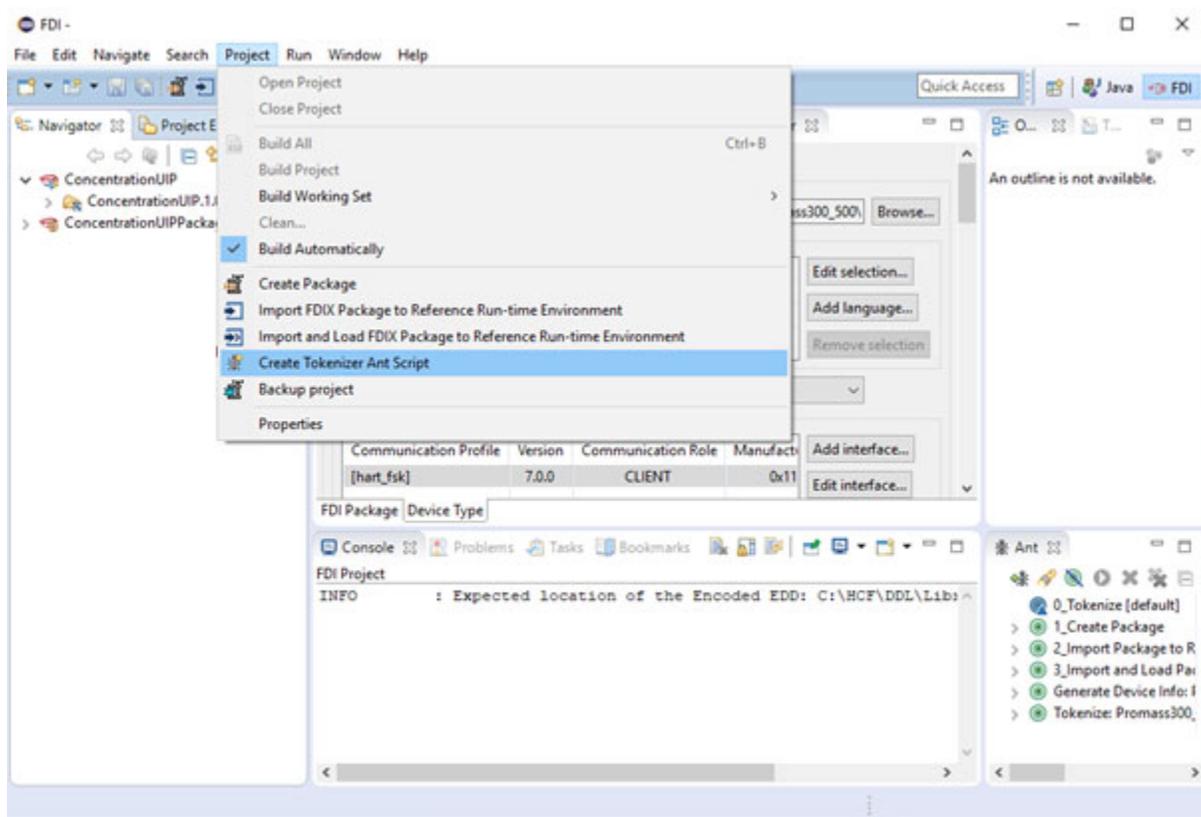


Figure 30. Create a Tokenizer Ant Script

## Testing the HTML5 UIP

Now you are ready to test your new HTML5 UIP with the FDI RRTE (Reference Run-time Environment) application.

For this you need to import the package into the RRTE and load it.

Triggering the “ConcentrationUIP” button in the Offline menu will show the HTML5 UIP in a new window. The example below shows the new window that displays the HTML5 UIP next to the RRTE.

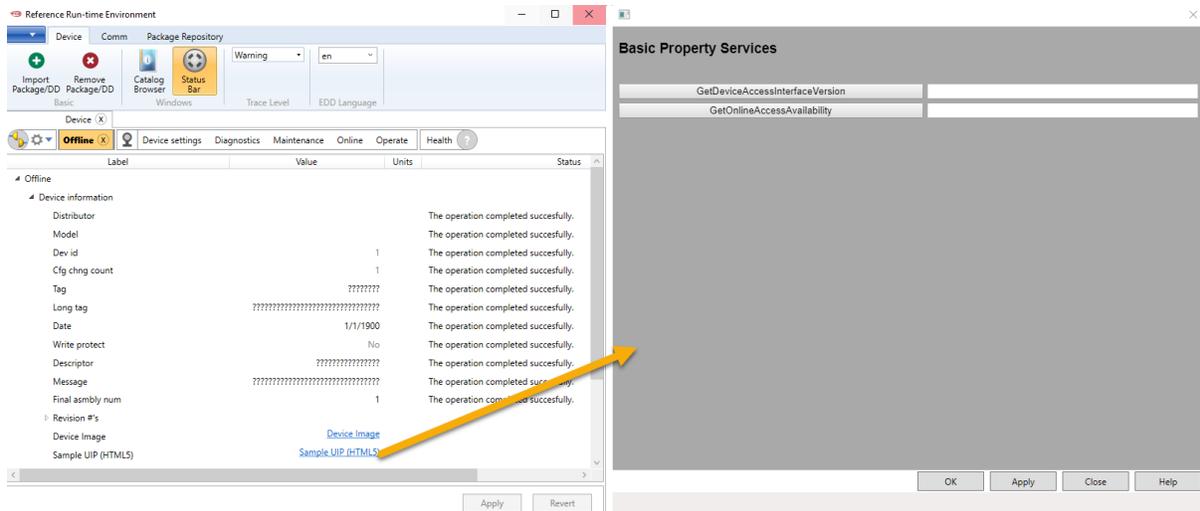


Figure 31. Trigger UIP from a menu

## How can you debug the UIP?

RRTE gives you an option to use the “Chrome developer tools” to test your code. To enable Chrome-DevTools you need to edit the file “C:\Program Files (x86)\FDI\Reference Run-time Environment\UipWebConfiguration.json” and change the value of “Environment” to “Development” and “ShowDevTools” to “true”.

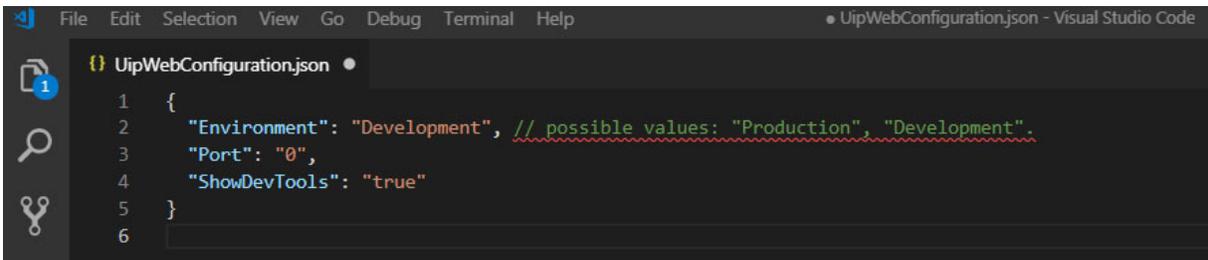


Figure 32. Enable debug support

Start the HTML5 UIP and a second browser window will show the Chrome-DevTools apps:

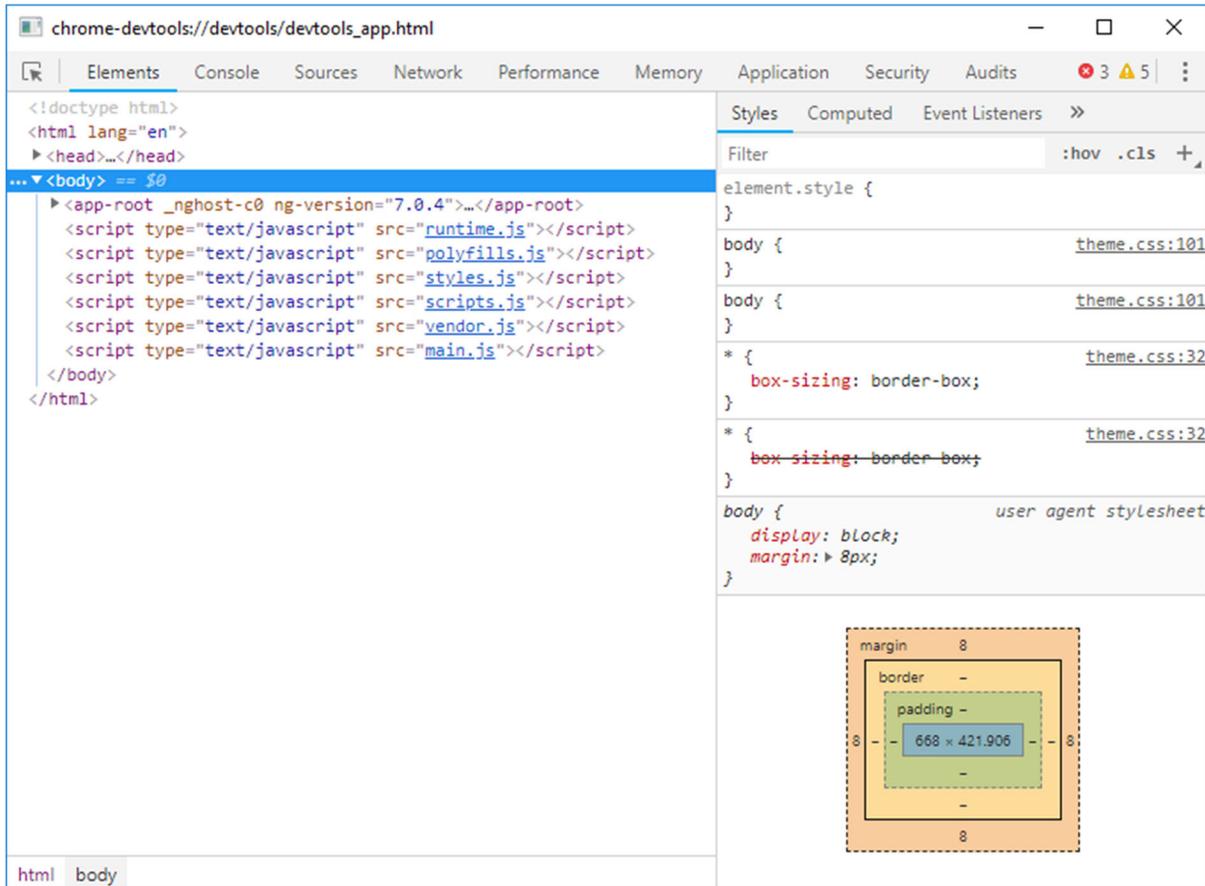


Figure 33. Chrome-DevTools give debugging support

After you have finished debugging, don't forget to switch the RRTE back to Production mode and run a final production test.

Everything fine? Congratulations! Now you can sign and submit the FDI Device Package to FieldComm Group for testing and registration.

## Implementing HTML5 UIPs on FDI Hosts

With the release of the FDI 1.2 Technical Specifications, the support of HTML5 UIPs is made mandatory for all FDI 1.2 hosts, the support for .NET UIPs is optional for a workstation host and not required at all for a mobile host.

This implies that in order to be FDI 1.2 compliant, FDI host vendors need to implement an HTML5 runtime into their products to run HTML5 UIPs.

There are two options for this:

- The host vendor decides to implement an HTML5 runtime on their own (see below an outline of the steps required)
- The host vendor uses the HTML5 runtime offered by FieldComm Group (FCG) – the UIEngine Common Host Component. The UIEngine is able to load and run HTML5 UIPs.

A possible architecture for implementing a runtime for HTML5 UIPs is shown below:

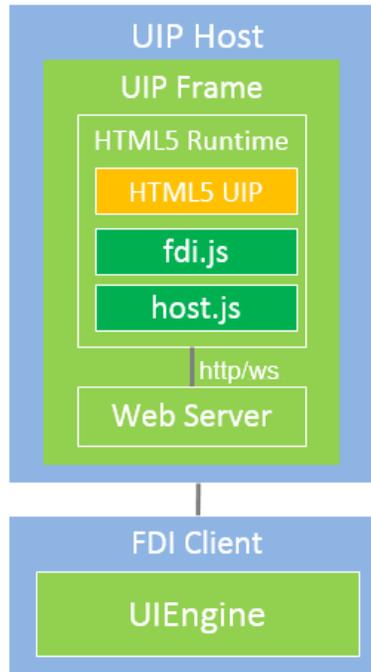


Figure 34. Runtime for HTML5 UIP

The implementation of an HTML5 runtime requires the following steps:

- Selection of a web browser engine. Popular candidates here are CEF (Chromium Embedded Framework), which is based on Chromium and WebKit. The web browser engine will run the HTML5 UIP provided by the device vendor. (The CHC UIEngine uses CEF.)
- Provision of a host application (UIP host) that is hosting the web browser engine in a UI frame. The UI frame is needed for the standard and specific action buttons defined by FDI.
- Providing an implementation for the UIP-FDI Client services. The FDI specification provides an interface description (FDI Type Library) of the UIP-FDI Client services in a TypeScript file: `fdi.ts`. There is no restriction from FDI on the technology that implements the services. Natural candidates here are web communication technologies, like REST APIs or WebSockets. CHC UIEngine uses REST and WebSocket communication and implements the FDI Type Library in the TypeScript file `host.ts`.
- In case of using web communication technologies, a web server should be implemented in the UIP host, to receive and serve the calls from the HTML5 UIP.
  - A strategy needs to be in place to start multiple UIPs at the same time. In the CHC UIEngine a separate web server process is started (as part of the UIP host) for each HTML5 UIP.
  - HTTP Port selection for the web communication is another important topic. The strategy could be either a set of fixed ports or a dynamic port assignment based on a search for a free port. The CHC UIEngine uses the latter approach.

- The HTML5 runtime should also give debugging support for developers. In the CHC UIEngine, HTML5 UIPs can be run in developer mode. In developer mode the Chromium developer tools are shown in a separate window helping to track calls and view traces and errors.

## Support and Resources offered by FieldComm Group

FieldComm Group provides resources for implementing and testing your FDI Device Packages, UIPs and hosts.

- FDI Technical Specifications are available via FieldComm Group's website.
- The FDI IDE helps to create and manage UIPs and FDI Device Packages. Developers can purchase the FDI IDE online from FCG.
- A Sample HTML5 UIP is included in the FDI IDE installation and gives you a jump start for developing your own UIP.
- Reference Runtime (RRTE) is a developer application included in the FDI IDE for testing the loading and execution of FDI Device Packages.
- The UIEngine is a component for running HTML5 UIPs and is part of the Reference Runtime (RRTE). It is part of the FDI Common Host Components (CHC) and can be used by host vendors to implement an HTML5 runtime within their applications. Host vendors may purchase a license for the FDI Common Host Components.
- The Host Conformance Test Kit provides a test suite to assist FDI Host developers as well as conformance test labs in verifying FDI Client conformance to FDI and EDDL specifications. A Reference HTML5 UIP is included in the Host Conformance Test Kit for testing the services of the FDI Type Library that is implemented by an FDI Client.
- The Device Package Conformance Test Tool (DPCTT) is the software tool that tests the conformance of the structure and signatures of an FDI Device Package. Required as a test for an FDI Device Package after the contents of the FDI Device Package have been tested with the Device. The DPCTT will also verify an HTML5 UIP within the FDI Device Package conforms to FDI specifications.
- Registration of tested and signed FDI Device Packages is offered by FieldComm Group. The registered FDI Device Packages are included in the Repository which is accessible from the Product Registry and the RRTE provides a connection to the online Repository. Developers can download and import FDI Device Packages directly into the RRTE.
- FieldComm Group offers Technical Support to get you started with the installation of new tools such as the FDI IDE. Visit the FieldComm Group Support Portal to read about developer tools, the registration programs, training or ask us a question.



**FIELD COMM GROUP™**  
*Connecting the World of  
Process Automation*

**FieldComm Group**

9430 Research Blvd., Suite 1-120

Austin, TX 78759 USA

Tel: +1 512 792 2300

[www.fieldcommgroup.org](http://www.fieldcommgroup.org)

Document Number: FCG AG10210, 1.0

January 2020

©2020 FieldComm Group