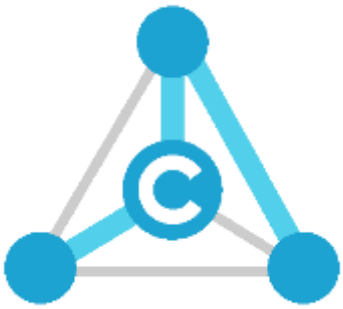
C-DEngine™ v. 4.0

Getting Started Guide



June 1st , 2017 – V4.006

Welcome to the C-DEngine™ V4.0 SDK

Thank you for using our C-DEngine V4.0 SDK – the rapid application development platform for IoT solutions.

With the C-Labs™ C-DEngine™ (pronounced “seed engine”), you can develop distributed solutions connecting local on premise devices with enterprise and cloud services. The C-DEngine automatically generates a modern Natural Machine Interface (NMI) that is compatible with any HTML5-enabled browser running on mobile devices such as tablets, pads, smart phones, and desktop browsers.

This Getting Started Guide walks you through the development of C# source code for a C-DEngine test hose, C-DEngine service running in a simple plugin, the routing of network traffic through the Cloud, as well as the creation of a more complex plug-in that accesses an Arduino board.

For support and more information, sign up for our forums at http://www.C-Labs.com/forums.

Have fun and be creative!

The Change History of the C-Labs C-DEngine,

has been moved to Appendix A

Chapter 1 - Getting Started

Supported Host Environments

The C-DEngine V4 SDK supports the following environments:

- Microsoft® Windows® XP or higher, including Windows® Vista, Windows® 7, Windows® 8, Windows® 10 in both 32-bit and 64-bit operation, using the .NET Framework 4.0 or 4.5.
- Linux® Operating System using Mono® 2.10 (or higher). We have tested Ubuntu®, Debian®, and Raspberry Pi™.
- Devices running Android™ version 4.0 or higher, using Mono® 2.10 (or higher) or Xamarin™.
- Apple® iPhone® running iOS® version 6.0 or higher, using Mono® 2.10 (or higher) or Xamarin™.

Development Systems

When targeting Microsoft Windows, developers will need a Windows workstation running one of the following development systems:

- Visual Studio® 2013 or later; Community, Professional, Premium, or Ultimate editions.

When using Xamarin Studio to create mobile phone software, your choice of operating system for your developer system impacts which mobile phone platforms you can target (although having two separate systems – one running Microsoft Windows and one running Apple OS X – allows you to target all Xamarin-compatible mobile platforms):

- A Windows® developer workstation can generate software for Android® or Windows® Phone.
- An OS X® developer workstation can generate software for Android® or iPhone®.

Compatible Browsers

The C-DEngine generates an HTML5 user-interface. Most recent browsers support HTML5 including Internet Explorer®, version 10 and 11, Microsoft Edge™, Opera, Firefox, Safari, Chrome and mobile browsers on iOS, Android 4.x, and Windows Phone 8/10.

Content of the SDK

The following resources are provided in the SDK:

Item	Description
Link to the Cloud Site:	https://cloud.c-labs.com
Installation Folder	**32-Bit System: c:\Program Files\C-Labs\C-DEngine SDK** **64-Bit Systems: c:\Program Files (x86)\C-Labs\C-DEngine SDK**
Visual Studio plugin with new project and new item templates	Contains a VSIX template for Host and Plugin-Service Development: • **CDEngineSDKTemplates.vsix** You can double click this file to add the Project Template to the Visual Studio
	The C-DEngine SDK ships with two files: (1) C-DEngine.dll – core engine library, and (2) CDMyNMIHtml5.dll – plugin to enable

Item	Description
C-DEngine Managed DLLs	browser-based user-interface featuring HTML 5. Seven folders provide versions of these files for different platforms: • **Android** – for use in Xamarin Studio and Managed C# Android projects • **IOS** – for use in Xamarin Studio and Managed C# iOS (iPhone/iPad) projects • **Mono** – for use in Xamarin Studio for Mono 4.0, 3.2.3, and 2.10. (For license details click here). • **NET40** – for use with .NET version 4.0 • **NET45** – for use with .NET version 4.5 • **W8U** – for use with Windows 8.1 Universal Store and Phone Apps. • **UWA\C-DEngine.dll** – for use with Windows Universal Platform Apps (Phone, IoT, Desktop, Xbox and HoloLens)
C-DEngine Documentation	http://www.C-Labs.com/docu
This Getting Started Guide	C-DEngine V4.0 - SDK - Getting Started.pdf

Figure 1 List of SDK Resources

The client engines (IOS/Android/Mini/W8U) are for end-nodes only and cannot relay C-DEngine telegrams. The full engines (Mono/.NET4x/UWP) can relay telegrams and should be used for nodes that relay telegrams across different networks. The UWP only supports local HTTP access but can relay to other nodes using WebSockets.

Installing template plugin for Visual Studio

Important! Make sure to install the Visual Studio templates that we provide in the SDK!
From the Windows Explorer, double click the file "**CDEngineSDKTemplates.VSIX**" in the installation folder. These templates enable Visual Studio 2013, Visual Studio 2015, or Visual Studio 2017 to generate the starting code needed to create C-DEngine components.

Chapter 2 - Creating your first IoT Solution

A C-DEngine IoT solution consists of at least two components:

1) A Relay Host Node – the main access point of information providing the NMI UX and data to all connected nodes. In many cases this is an application you already have and want to extend to the IoT.
2) A device or service plugin – a DLL that contains the business logic and device access code for each device type that wants to participate in the IoT solution. This plugin is where all the device specific code will be written. After you created the host once, you will only work on this plugin.

The third component – a browser based dashboard and portal is automatically generated by the relay host node and does not require any additional coding.

Optionally the IoT solution can be extended to mobile devices on the internet by adding a cloud-relay. This evaluation version of the C-DEngine uses a shared cloud-relay node.

Component 1: Creating a C-DEngine Host-Relay (Debug) App

The C-DEngine is a DLL that manages the IoT communication and NMI (Natural Machine Interface) elements of "Things". It can run in any type of hosting process such as Windows Services, Win Form Apps, Console Application, Store App and even ASP.NET Web-Application.

One node can relay information to any other node. The host we are creating here will be able to relay information from locally attached device to a cloud node that than relays the data to a mobile device. Local mobile devices can talk directly to this host-node without the need to connect to the cloud/internet.

For our Getting Started we are creating a Console Type application. The C-DEngine Visual Studio Templates contains a template for this type of host.

> If you want to use the fast WebSocket protocol under Windows 8/Server 2012 (or later), you must launch Visual Studio "as Administrator…" and use the C-DEngine for .NET 4.5.
>
> All other supported OS can run WebSockets without Admin privileges – but uses a slightly slower "WebSocketsSharp" stack (licensed under MIT see http://sta.github.io/websocket-sharp/)

Creating your first C-DEngine Test-Host in 7 easy steps

1	Create a new Project of type "cdeHostProject" (one of our Visual Studio Templates) and name it "MyTestHost" (the specific name is not important, although we refer to this name later).

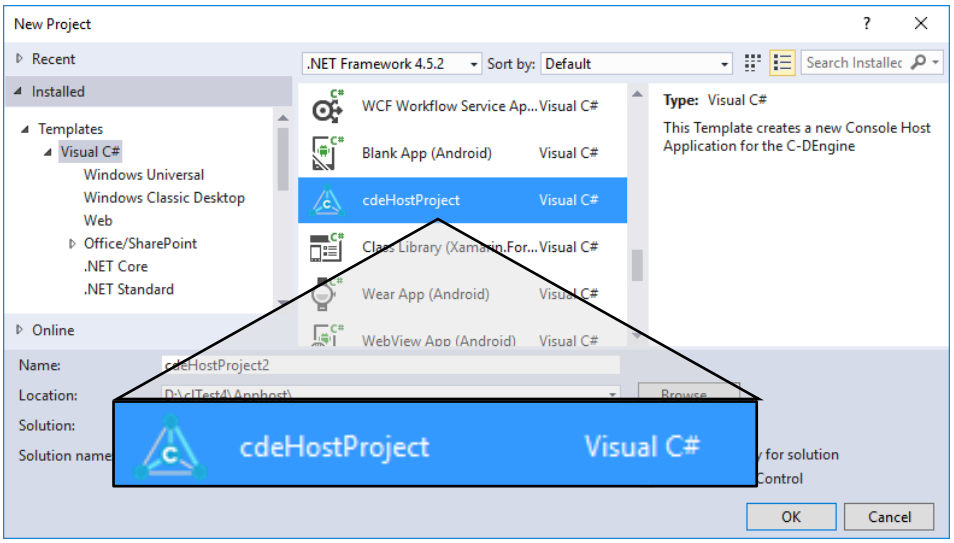


Figure 2 Creating a new Project of type "cdeHostProject"

2	Add a reference to this library, **C-DEngine.dll**, which is the core engine.

In the Visual Studio Solution Explorer Window, right-click the **References** item. Select **Add Reference…** on the popup menu. Browse to select the compatible files **C-DEngine.dll** and **CDMyNMIHtml5.dll.** The default version of the .NET Framework, for Visual Studio 2013 and later, is .NET Framework 4.5. (On 32-bit systems, the library folders are in **c:\Program Files\C-Labs**. On 64-bit systems, the library folders are in **c:\Program Files (x86)\C-Labs**.)

3	In the file "program.cs", at line 24, replace "<<CREATE WITH GUID TOOL>>" with a valid GUID. Create a GUID in Visual Studio by clicking **Create GUID** in the Visual Studio **Tools** menu. Use the "Registry Format".

The essence of creating a C-DEngine Host Application is to specify some basic parameters and start the C-DEngine (which occurs on line 63 with the call to `MyBaseApplication.StartBaseApplication`). Source code comments provide some guidance on details of specific parameters.

4	You can now build and run the Host Application.

If you have IIS or any other web server installed that is using port 80, you can change the port in Line 39/40 to any other port you like. We recommend using some port around 8700 as UPnP on Windows requires a port >4096 to show in the Windows Explorer.

You will see an error message in the command line that the Communication could not be started.

If you are using an older Version of Windows/Server than Windows 8/Server 2012, you must change the WebSockets port (MyStationWSPort) to something else then 80. (i.e. 81 or 8080).

You must point your browser at the corresponding address (I.e.: http://localhost:8700/nmiportal)

5	To test that the Host is running properly open your Internet browser and point it at http://localhost/nmiportal.

You should see the welcome screen of “My-Relay”.

> For security reason, if you have not launched Visual Studio “as Administrator” you will not be able to use “localhost”. You must replace LOCALHOST with the full DNS name of your PC. This allows you to login to your NMIPORTAL using REST.

The host application creates a random Scope ID in line 59.

6	You can see your “Current Scope:” in the first line of debug output of your Host Application

7	You can enter this ID in the Welcome screen to log in into your Relay.

For debugging, you can use a fixed ScopeID instead of the “GenerateNewScopeID()” method. Just use any string with at least an 8 letters/numbers combination. Sometimes the browser is caching your page. You will not be able to login if that happens. Simply refresh your browser and it should work.

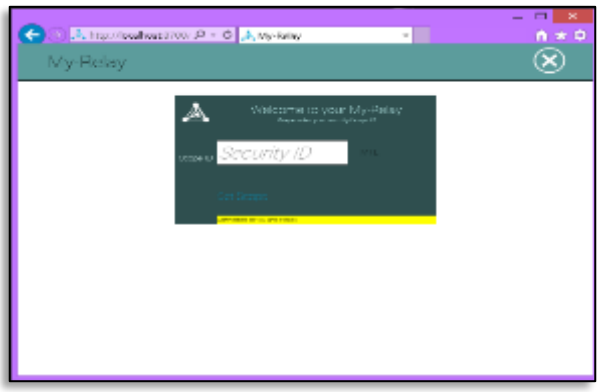


Figure 3 Welcome to your My-Relay screen

> If you change “UseRandomDeviceID” to false, the ScopeID and the Port settings will be persisted and cannot be changed anymore without erasing the storage files in “ClientBin/Cache”.

If the yellow line says, “Connected to WS and Ready” you have full support for WebSockets. If it only shows “REST” you have access via REST. The speed drops back to REST if the WebSockets could not be initialized on any of these conditions:

- You set the MyStationWSPort to zero (turning off the WebSockets on purpose)
- WebSockets cannot be supported on your platform. Please report your platform to C-Labs that we can investigate (send email to support@C-Labs.com)

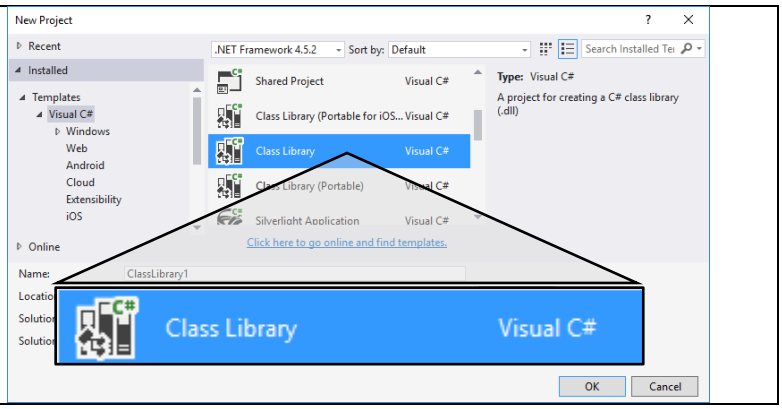
Congratulations! You have your first C-DEngine Test-Host running!

You can log in with the ScopeID you can find in Step 6 and click on the NMI Admin button to get to the NMI Administrator Dashboard. This dashboard is explained in chapter 4.

Component 2: Creating a C-DEngine Plugin

Now that you have a C-DEngine Host Application, you want to write your own custom plugin that reads data from an arbitrary device and sends this data securely to connected mobile devices or browsers.

Follow these steps to add a new Plugin-Service, as a Visual Studio project:

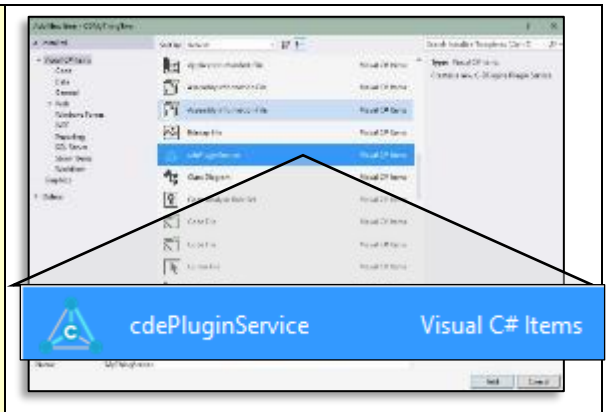
1	Under File menu, select “Add” and “New Project”. In the Next Project dialog, select “Visual C#”, ”Class Library”, as shown in the diagram. Then, name your project “CDMyThing” (pronounced “seed my thing”).	

Note: A Plugin must have a file name that starts with “CDMy”, “C-DMy”, or the file extension must be “.cdl”. For debugging, I recommend staying with CDMy prefix. Example: “CDMyCamera”

2	Add a reference to two libraries in your newly-created project: **C-DEngine.dll** (the core engine) and **CDMyNMIHtml5.dll** (The HTML 5 user-interface plugin). At this point, you have a single solution that contains two projects. Select a version of C-DEngine.dll to match your project needs. (On 32-Bit Systems, look in c:\Program Files\C-Labs\C-DEngine SDK and for 64-Bit Systems, look here: c:\Program Files (x86)\C-Labs\C-DEngine SDK.

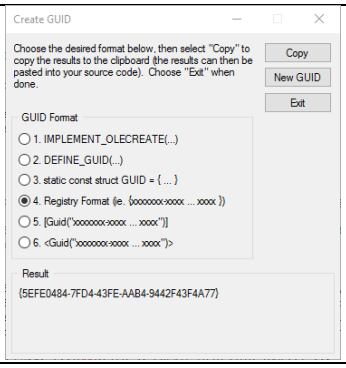
IMPORTANT: The Host must have a reference to the same or higher .NET version of the C-DEngine

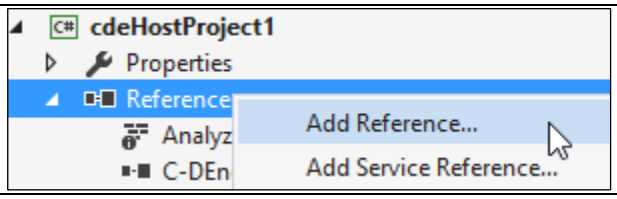
3	Delete the default “Class1.cs” file

4	Add a new Item with the Visual Studio command Project | Add New Item… For a template, pick “cdePluginService. Rename it to “MyThingService.cs”. Any name works. We recommend using a recognizable name representing the “Thing” you want this service to manage. Replace “Thing” with that name (i.e. “MyArduinoService”).	 ***Figure 4 Adding a new C# item***

5	Now open the file “MyThingService.cs” and go through the TODO Steps outlined in the source:

6	In Line 34, give your service a friendly name.

7	Use the “Create GUID” tool, select the registry format and hit “New GUID” than click on “Copy” and in line 37 replace `<<TODO:STEP 2: CREATE WITH GUID TOOL>>` with the new GUID.	

8	In the Visual Studio Solution Explorer, select the References item in your host application project. Right-click and select **Add Reference…** from the popup menu.	

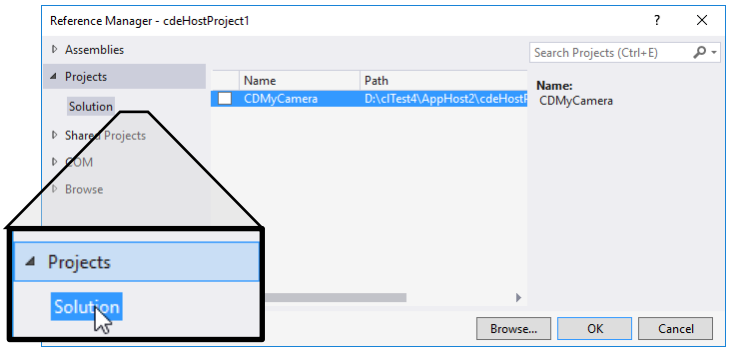


Figure 5 Adding a reference screen

In the **Reference Manager** dialog, select **Projects | Solutions** on the left panel.

Note: Plugin references help when debugging, but otherwise are not needed. In production, a C-DEngine host application uses all properly named plugins located in the host application’s execution folder.

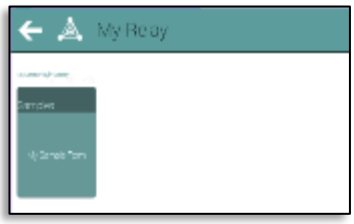
9	Build the plugin, and test it by running the MyTestHost Application.

10	Start a HTML5 capable browser and navigate to http://locahost/nmiportal (don’t forget the port w not using Port 80).

11	Login with the ScopeID from the previous chapter.

After successful login you will see:

Click on “My Demo Plugin Screens” to see:



Click on “My Sample Form” to see:

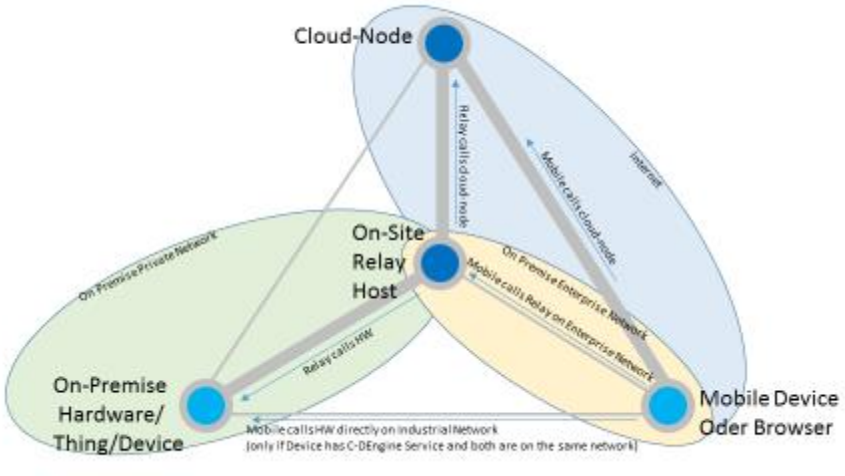
12	Enter values in the Sample field and see how the bar changes. Click on the Bar and see how the displayed value changes.

All code needed to distribute your “Thing” to mobile devices and browsers will go in the CDMyThing plugin project.

Congratulations! You built your first C-DEngine IoT Service.

Component 3: Accessing you Relay-Host via the Cloud

The final step of the getting started is to connect our relay to the cloud.



In this image you can see the On-Site Relay-Host in the center of the communication flow.

It hosts all the business logic of a C-DEngine IoT solution. The Mobile device (to the right) can access the relay-host directly if it is connected to the on- premise enterprise network. In smaller environments, such as in a home, this might be the same network as your devices are connected to ("on premise private network").

Your mobile device must go through the cloud to connect to your relay-host only if it's not on the private network.

Let's add the cloud-node to our IoT solution.

For the evaluation of the C-DEngine, C-Labs created a shared cloud node that you can use for testing.

First check out the cloud portal to see that you do not have an active cloud connection right now.

1	Open your favorite (HTML5) browser and go to https://cloud.c-labs.com/nmiportal

You will see the same login screen you saw on your local relay. But if you type in your scope ID it will say "Login Failed".

You must point your relay at the cloud first.

2	In your MyTestHost project open "program.cs" and insert a new line under line 40 and type CloudServiceRoute="https://cloud.c-labs.com" If you have support for WebSockets you can replace https with wss (WebSocketsSecure) for even faster speed.

3	Build and run your test application.

4	Now refresh your browser and type in your ScopeID. You should see the cloud dashboard including your "My Demo Plugin Screens". The dashboard also includes plugins that run in the cloud.

Figure 6 Cloud Dashboard in cloud

For the C-DEngine evaluation C-Labs added the “Computer Vitals”, “Feedback” and “VThings” plugin. For a detailed description of these plugins please visit our forums at http://www.C-Labs.com/forums.

Only the Cloud-Node owner can upload plugins to the cloud!

If you log-in to your local relay again, you will also see the cloud plugins showing in your dashboard. The C-DEngine is consolidating the UX from all known nodes with the same security context and displays the result in the browser.

Open two browsers, one pointing at your local relay-host and one at the cloud-node. Then go to your plugin screen and enter a value in the input field or click on the bar. You should the new value changing on both screens.

Chapter 3 - Developing a complex sample Plugin in C#

Arduino boards are selling fast and making headway in the IoT.

To show how easy a plugin can be developed, we are building a plugin for a standard Arduino board.

As pre-requisites, this chapter assumes you have the Arduino 1.0.5 IDE installed and your board is connected to your PC via USB (which internally is just a virtual COM port). You can download the IDE from (http://arduino.cc).

The chapter also assumes the Arduino board is connected to COM Port 3 (COM3). If you want to find out the COM port of your Arduino board, refer to the manual or Arduino Software.

We are also assuming you connected an analog sensor to pin A1 (we are using a Photo Sensor from the SUNFOUNDER Sensor Kit).

1	To work with our plugin we have to download a very small program in the Arduino board:

```
const int photoPin=A1;
const int led = 13;

void setup()
{
  Serial.begin(115200);
  pinMode(led, OUTPUT);
}

char incomingChar;
boolean isHigh=false;
void loop()
{
    for (byte count = 10; count > 0; --count) {
      if(Serial.available() > 0){
          incomingChar = Serial.read();
          //your protocol goes here
          switch (incomingChar)
          {
              case 'L':
                  digitalWrite(led,LOW);
                  break;
              case 'H':
                  digitalWrite(led,HIGH);
                    break;
          }
      }
      Serial.print("Photo:");
      Serial.print(analogRead(photoPin),DEC);
      Serial.println(":");
```

```
      Serial.print("Counter:");
      Serial.print(count-1);
      Serial.println(":");
      delay(50);
    }
}
```

This code reads from the serial port and if an “L” was sent, turns off the onboard LED and an “H” turns it back on.

It also reads the (Photo) sensor and send its value as well as a counter value via serial in the format:

```
Photo:<value>:\nCounter:<count>:\n
```

Now let’s write the C-DEngine plugin for this Arduino program:

1 Add the following block of code right behind the “Init()” method in the MyThingService.cs:

```
SerialPort currentPort; // Declares the COM Port

    void currentPort_DataReceived(object sender,
                                  SerialDataReceivedEventArgs e)
    {
        SerialPort sp = (SerialPort)sender;
        string indata = sp.ReadExisting();

        string[] tArgs = indata.Split('\n');
        foreach (string t in tArgs)
        {
            string[] tCmd = t.Split(':');
            if (tCmd.Length > 2 && tCmd[1].Length > 0)
            {
                string val = tCmd[1];
                double dVal = TheCommonUtils.CDbl(val);
                switch (tCmd[0])
                {
                    case "Counter":
                        TheThing.SetSafePropertyString(MyBaseThing,
                        "Counter", val);
                        break;
                    case "Photo":
                        TheThing.SetSafePropertyString(MyBaseThing,
                           "PhotoSensor", val);
                            break;
                }
            }
        }
    }
```

2	Resolve the “SerialPort” with “System.IO.Ports”. The Port configuration is for the Arduino Uno R3.

The event-handler read the current COM buffer and splits the received string first by linefeed (“\n”) and then by colon (“:”). The switch then looks at the first command parameter and if it is “Counter” sets the property “Counter” to the new value. Then does the same for the “PhotoSensor” value.

Remark: As the COM receiving buffer might not deliver all characters properly, you might lose some characters. To do this properly you must do some better COM Buffer processing. We omitted this here to keep the sample simple.

3	Locate line 116 (TheThing.SetSafePropertyString…”Sample”…) in the Init() function and insert the following code after this line:

```
TheThing.SetSafePropertyString(MyBaseThing, "PhotoSensor", "0");
string port = MyBaseThing.Address;
if (string.IsNullOrEmpty(port))
   {
      port = "COM3";
      TheThing.SetSafePropertyString(MyBaseThing,"Address",port);
   }

   TheThing.SetSafePropertyBool(MyBaseThing, "IsConnected", false);
   try
   {
      currentPort = new SerialPort(port, 115200);
      currentPort.DataReceived += currentPort_DataReceived;
      currentPort.Open();
      TheThing.SetSafePropertyBool(MyBaseThing, "IsConnected", true);
   }
   catch { }
```

The first new line declares a property “PhotoSensor” in the plugin and initializes it with zero.

The next couple of lines define the port we are using to connect to the Arduino board. For the sake of keeping it simple, we hardcode the port here. You can change the port later with the NMI UX.

The last block starts and opens a serial port and sets the “IsConnected” property to true, if the port was successfully opened.

The “Init()” method of each plugin is called by the C-DEngine when all engines are started and the internal “ThingService” is ready and operational.

Next, we need to declare the “currentPort” variable and create an event-handler for the “DataReceived” event of the COM port. If you have a different Arduino Board, you might have to change the parameters of the ComPort.

For the Leonardo board add these lines before “currentPort.Open()”:

```
currentPort.Handshake = Handshake.None;
currentPort.RtsEnable = true;
currentPort.DtrEnable = true;
```

Now we have the values in the plugin and can create the NMI UX for it

3 In the “CreateUX()” function add the following lines behind the AddSmartControl(…”Sample”,”Sample”) line:

```
TheNMIEngine.AddSmartControl(MyBaseThing, tMyForm,
    eFieldType.SingleCheck, 4, 0, 0,
    "Is Arduino Connected", "IsConnected");
TheNMIEngine.AddSmartControl(MyBaseThing, tMyForm,
    eFieldType.Number, 5, 0, 0, "Photo Sensor", "PhotoSensor");
TheNMIEngine.AddSmartControl(MyBaseThing, tMyForm,
    eFieldType.SingleEnded, 6, 2, 0, "COM Port", "Address");

ThePropertyBag PropertyBag = new ThePropertyBag()
    {"MaxValue=10",
      "TileWidth=6",
      "TileHeight=2",
      "IsHorizontal"};

TheNMIEngine.AddSmartControl(MyBaseThing, tMyForm,
    eFieldType.BarChart, 7, 0, 0, "Counter", "Counter", PropertyBag);
```

This code will create new controls on our Sample Form for the Arduino sensors.

The first line adds a “Checkbox” (eFieldType.SingleCheck) for the “IsConnected” property. If your Arduino board connected successfully, you will see this checkbox selected.

The next line adds a “numeric output” field for the “PhotoSensor” property and the following line a “single line text output” for the current COM port.

Address (such as “FriendlyName” and “LastUpdate”) is an intrinsic property of every “Thing” (located in MyBaseThing) and managed by the C-DEngine. It should be used for any direct connection parameter to a device. It can hold any arbitrary addressing string.

The last two lines add a single bar-chart control to the form that will display the “Counter” property.

The PropertyBag is a common way for the C-DEngine to set custom properties. The BarChart has a couple custom properties such as "MaxValue" and "IsHorizontal". Most other controls have the "TileHeight" and "TileWidth" property to define the size of the control in "Tile" units. By default one Tile-Unit is 73x73 pixels.

If you run your test host now you should see the following Form on your "Sample" page:

Figure 7 Demo Page Screen

The bar should move from right to left, updating every 50ms. If you are connected with REST, you will see this bar move much slower.

Now let's add some buttons to send the "L" and "H" to the Arduino board.

4 Add the following lines right below the "AddSmartControl(…BarChart)":

```
PropertyBag = new ThePropertyBag() { "PreventDefault", "TileWidth=4", "TileHeight=2" };
TheFieldInfo mTapButton = TheNMIEngine.AddSmartControl(MyBaseThing, tMyForm,
       eFieldType.TileButton, 33, 2, 0, "Tap LED", false, "", null, PropertyBag);
mTapButton.RegisterUXEvent(MyBaseThing, eUXEvents.OnClick, "LED_ON", sinkMe);
TheFieldInfo mTapButton2 = TheNMIEngine.AddSmartControl(MyBaseThing, tMyForm,
       eFieldType.TileButton, 34, 2, 0, "Tap LED OFF", false, "", null, PropertyBag);
mTapButton2.RegisterUXEvent(MyBaseThing, eUXEvents.OnClick, "LED_OFF", sinkMe);
```

The first line defines a PropertyBag that sets the TileHeight and TileWidth of a button to a larger button.

AddSmartControl() returns a "TheFieldInfo" class. This class has a method to register an EventHandler for various UX Events.

We are using the "OnClick" event in order to respond to somebody touching/clicking the button.

5 Add the Event Hander for these OnClick events right after the "CreateUX()" method:

```
        void sinkMe(ICDEThing pThing, object Details)
        {
            if (currentPort == null) return;
              TheProcessMessage pMSG = Details as TheProcessMessage;
```

```
            if (pMSG == null) return;
            string[] cmd = pMSG.Message.PLS.Split(':');
        switch (cmd[1])
        {
            case "LED_ON":
                currentPort.Write("H");
                break;

            case "LED_OFF":
                currentPort.Write("L");
                break;
        }
    }
```

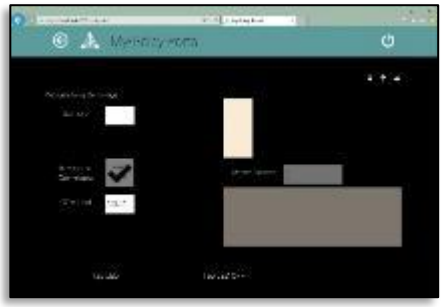
An OnClick UX Event handler returns a TSM (The System Message) and in its PLS (Payload String) parameter you find the details with the syntax:

<Amount Of Touchpoints>:<Cookie handed in during the event registration>:<cdeMID ID of the Record in the table this object is reference in>

In our case, we added "LED_ON" and "LED_OFF" to our buttons as cookies.

6	Let's run your app now and see what is happening. You should see the following screen:

If you touch the "Tap LED" button your LED on your Arduino board should go on. And if you touch the "Tap LED OFF" button, your LED should go off.



You might be wondering why the COM port is not stored when you change it in the UX. The reason relates to the "UseRandomDeviceID" setting in the Host application. If this is "true" your Host-Relay will not store any value on disk and every time you restart your host application the "Address" is empty.

Figure 8 My-Relay Demo Page

7	Just change this value in the program.cs to "False" and you will see the Host-Relay will remember the COM port you entered.

Login with the cloud portal (https://cloud.c-labs.com/nmiportal) and check if you see the same screen and can do the same interactions from there. You will notice if you keep the cloud portal open and terminate your Host-Relay the counter will stop but your browser session does not end. And if you restart your Host-Relay, the cloud picks up again and continues to show the counter.

Again this only works with "UseRandomDeviceID" is set to "false".

We hope you could see how powerful and simple you can create a distributed IoT solution with the C-DEngine.

As next steps with your plugin, try to add a “Connect” button that allows you to reconnect to a COM port if you changed the COM port address.

Congratulations! You built your first IoT Solution.

Appendix A – Change History

Version 4.006 – June 1st, 2017

- New NMI TypeScript files with new definition file versions cdeCore-4.0.0.d.ts, cdeNMI-4.0.0.d.ts and cdeNMIBaseControls-4.0.0.d.ts
- New ISBConnect Class that allows to initiate a connection to a node with a different Scope for Scope-Federation.
- New Experimental ISOlater for Plugins
- Updated SDK Templates

Version 4.004 – April 4th, 2017

- ADFS/OAuth2 Interfaces ready. A new Plugin "OpenID" can be used to connect to an ADFS/OAuth2 login provider
- Thing Historian improvements
- New "TheISBConnect" class to create custom scoped ISB Connections (Multi-Scoping and Federation)

NMI Enhancements:

- NMI Performance improvements
    - dynamic loading of screen
    - Mini-Subscriptions for Fields and ThingsIDs on First Node to reduce traffic to JavaScript nodes
- Clock on Larger screens (all but Mobile)
- Full support for all Windows 10 platforms (Desktop, Mobile, HoloLens, X-Box, and IoT). "NMI Viewer" app is ready for the Windows Store
- New "PlatformBag" on Forms (TheFormInfo) and Fields (TheFieldInfo) allows to set platform (Desktop, Mobile, HoloLens,X-Box and IoT) specific properties.
- New SystemLog viewer for all nodes in a mesh in "NMI Admin" section (administrator only)
- New option to disable plugins from loading
- New TXT=NMI_ALERT:<timeout> message – displays a bigger toast to announce new devices
- New SetUXProperty for Table Fields. New Syntax adds a ThingID as the 4th parameter to precisely point at a Row/Field in a table
- Enhanced Security around JavaScript

Version 4.001 – January 2017

Major Enhancements:

- Route Optimization by reversing ORG to GRO as the route for optimized travel time
- Properties of Properties: Properties can be annotated with properties for hierarchically description of Tags. New Syntax for sub properties is "[Level1Property].[Level2Property]…". The NMI allows to specify these hierarchies in the "OnUpdateName".
- Globalization in 5 languages (English – the default, German, Spanish, French, Italian). Language can be selected via the LCID in TheSessionState and the NMI User Manager settings

Version 3.220 – November 10th, 2016

- NMI Improvements
    - Version 4 preview: New TileFactorX/Y – allows to specify a divisor that will be used to calculate the final grid size of a control. i.e: TileFactorY=2 means that TileHeight=1 is resulting in TileHeight=1/2 = 0.5. This allows for smaller Controls. Attention: in V3.220 only TileGroup, CollapsibleGroup, TileButton, SmartLabel and StatusLight are supported
    - New Support for Image links in FacePlates using the macro "<%I:propertyName%>"
- New Support for DeepLinks in the cloud using /nmiportal?CDEDL=<PluginGuid>;<ScreenGuid>.

Version 3.219 -October 16th, 2016

- SetVersion in InitEngineAssets is no longer needed if the "AssemblyFileVersion" in the AssemblyInfo.cs is set correctly.
- New eThingEvent: OnPropertyChangedByUX: Fires only when the property was changed via the NMI not any backend acitivities.
- AddSmartControl has an event sink for a function that was called OnPropertyChange of the property declared in "OnUpdateName". Starting 3.219 this event is using the new "OnPropertyChangeByUX"

Version 3.218 – October 10th, 2016

- Internal improvements
- New NMI "HidePins" property on Dashboards and Forms

Version 3.215 – September 1st ,2016

NMI Internal improvements

- Improved Global Thing management
    - TheThingRegistry.UnregisterThingGlobally() to unregister a Thing from Global Events
- New MyServiceHostInfo.Coufs – Create User On First Start to autogenerate Users for a Relay Node. This only works on blank and brand new nodes

Version 3.213 – August 22nd, 2016

- NMI Additions
    - All controls now supporting "OnKeyDown" and "OnKeyUp" to hook keyboard input.
- All Resources on Mono (Linux) in the file system have to be lower case. /ClientBin will be renamed to /clientbin. If you use File.IO API Calls please use "TheCommonUtils.cdeFixupFilename()" to create the correct path under the /clientbin folder (correcting the paths for AWS, Azure, Windows and Mono).

Version 3.208 – July 1st, 2016

BREAKING CHANGES:

- Plug-ins must be signed, or signature validation must be turned off in app.config (dev only, not supported in production "`DontVerifyTrust=true`"

    - In order to increase code-signing verification "`VerifyTrustPath=true`" can be set to validate the complete code-signing certificate path. This REQUIRES an active internet connection.
- On Windows 7 signed Services can delay the startup beyond the point of proper starting. We recommend that you put the following lines in your App.config:
    - <runtime>
    - <generatePublisherEvidence enabled="false" />
    - </runtime>
- TheServiceHostInfo.MyServiceURL renamed to TheServiceHostInfo.LocalServiceRoute
- Changes to ICDEThing Interface:
    - Init(): must now indicate completion (return value true or FireEvent(eThingEvents.Initialized, true))
    - CreateUX(): must now indicate completion (return value true)
    - Delete(): new method called when thing is unregistered permanently. Disconnect, release or delete any resource including any other thing instances created and owner by this thing
    - HandleMessage(): now has an additional sender parameter, to be in line with .Net event handler patterns
- Plug-in Version must be 3.200 or higher
- TheThingRegistry.RegisterThing now returns null in more cases than before:
    - Things without an activated license or entitlement will not be registered.
    - If a BaseEngine was not specified on a TheThing, it will register but cannot be licensed.
- NMI Changes
    - Form Views now encore Labels in front of controls. These labels can be turned off using the Property "NoTE=true".
    - FldWidth is now the TileWidth for fields in Tables. This was used in 3.1 as well but not consequently enforces as in 3.2

Version 3.200 – June 5th, 2016

BREAKING CHANGES:

- MyBaseEngine does not longer Support "GetBaseThing().GetBaseThing()" to get to its ICDEInterface. Now just use "GetBaseInterface()"
- Rename of "ctrlReveilButton" to "ctrlRevealButton"

- New NMI Properties on ctrlTileGroup and ctrlCollapsibleGroup: IsHScrollable and IsVScrollable
    - Allows to make a TileGroup or CollapsibleGroup horizontally or vertically scrollable

Version 3.125 – May 23rd, 2016

- Minor bug fixes for faster Shutdown and NMI Portal Setting

Version 3.124 – April 10th, 2016

- New APIs for Probing of Engine/Service status:

- **TheThingRegistry.IsEngineStarted(<EngineName>,<AllowRemote) –** returns true if the Engine has been started by TheEngineHost. It has NOT Initialized but it is considered a live Engine on the current node
- **TheThingRegistry.IsEngineInitialized (<EngineName>,<AllowRemote)** – returns true if the Engine has been Initialized (SetInitialized() was called as part of "ProcessInitialized()". At this point the Engine/Service is fully operational.

Version 3.123 – April 5th, 2016

- New "Updater" functions on cdeP
  - **SetUpdater(<guid>)** - sets a Guid of TheThing that has updated the property
  - **GetUpdater()** return the Guid – TheThing can test in an OnPropertyChange event if it was the setter and if the setter and the OnPropertyChange receiver are the same, NOT act on the change event
  - **ResetUpdater()** – Resets the Guid for the next setter
- For security reasons unscoped Relays are no longer allowed to connect to CloudServiceRoutes
- New "Tag" Property on NMI Controls allowing to set a cookie with any NMI Control
- Fix for dots (.) in Property Names

Version 3.122 – March 29th, 2016

- Improvement for ISB Path Management to allow legal and wanted MITM Gateways
- Improved cdeStatus.aspx page
- Improved Subscription Management in Cloud

Version 3.121 – March 23rd, 2016

- Improved PubSub – DirectPub to Node not going over complete mesh once delivered
- CDE_SYSTEMWIDE now bound to SCOPEID in Mesh
- cdeNMI improvements for Float and HorizontalAlignment
- No API Changes

Version 3.120 – March 14th ,2016

- BREAKING CHANGE:
  - The engine now sends different SetProperty calls to clients for SetUXProperty (now sends "SETNP") and Set(Thing)Property (continues to send "SETP").
- New and much improved NMI Controls management increasing the UX performance in browsers
- New WebSocketSharp Stack for better Firefox compatibility on Windows 7 and XP
- Lots of reference documentation updates on http://www.C-Labs.com/docu

Version 3.118 – February 29, 2016

- New TheThing Throttling API to reduce excessive property changes being sent to clients
- Many performance enhancements
- More reference documentation

Version 3.117 – February 21, 2016

- Breaking API Changes:

  ```
  TheBaseAssets.MyServiceHostInfo.IgnoreAdminCheck = true, (was “IgnoreAdmin”)
  ```

  New UseRandomDeviceID behavior:

  ```
  TheBaseAssets.MyServiceHostInfo.UseRandomDeviceID
  ```

  Can no longer be set by the Host or plugins and has to be set at Startup by either adding to the App.Config:

  Or adding the setting to the ArgList (command line parameter):

  ```
  ArgList.Add("UseRandomDeviceID", "true");
  ```

Version 3.115 – February 1st, 2016

- Life FacePlates on TheDashPanelInfo
- Improved startup sequence of relay
- Stability improvements
- New APIs to enable the publishing events on properties: TheThing.SetPropertyEvents(pName, true|false);

Version 3.114 – January 21th, 2016

- New FacePlate Support for TheDashPanelInfo and TileButtons.
- New ctrlFacePlate for complex HMTL5 based face plates
- Support for Windows UAP applications
- New templates for “cdeServiceWithThings”

Version 3.113 – January 6 th, 2016

- Improved NMI compatibility between JavaScript and C# plugin

Version 3.111 – December 5th, 2015

- New improved APIs for NMI
- Synchronized with Factory-Relay V1.111
- Bug Fixes in cdeNMI JavaScript engine
- New ThePropertyBag class for dynamic control properties
- New nmiCtrlxxx classes for Control Property Bags
- Removed references to C-Labs LLC (now C-Labs Corporation)

Version 3.084 – June 7th, 2015

- Improvements for Connection with Cloud
- Security enhancements
- UPnP Enhancements

Version 3.083 – June 2nd, 2015

- New “SecureProperties”: Properties and NMI Controls marked as secure are stored encrypted automatically. To Declare a secure property use:

```
<TheThing>.DeclareSecureProperty(“<propertyName>”);
```

  Any NMI control that is set using the control type “ctrlPassword” will automatically create a secure property for the OnUpdateValue

Appendix B - The NMI Administration Portal

Natural Machine Interface ...the evolution from HMI (Human Machine Interface) adding modern user experiences such as Gestures, Voice input, Touch, and other NUI technologies

You can find an overview in the tutorial for the C-Labs Natural Machine Interface:
http://www.c-labs.com/docu/001-NMI%20Tutorial.pdf

Usage Rights and License

These are the basic rights, permissions, and license restrictions of the C-DEngine components:

- This evaluation version allows you to build test applications using the C-DEngine components.
- You are not allowed to distribute any application using the C-DEngine without a valid signed license agreement or written permission by C-Labs Corporation.
- You are not allowed to reflect, disassemble, or otherwise reverse engineer the C-DEngine components
- The C-DEngine runtime remain property of C-Labs Corporation.
- The C-DEngine, Factory-Relay and Home-Relay are Copyrights © of C-Labs Corporation.

Support

If you have questions, suggestions, or other feedback, please send them by email to mailto:support@c-labs.com

For support and news about the C-DEngine visit: http://www.c-labs.com/c-dengine

For online reference documentation look at http://www.C-Labs.com/docu

You can reach us on

- LinkedIn: http://www.linkedin.com/company/c-labs-llc
- Twitter: http://twitter.com/clabsglobal
- Facebook: http://www.facebook.com/pages/C-Labs-LLC/149850798454900