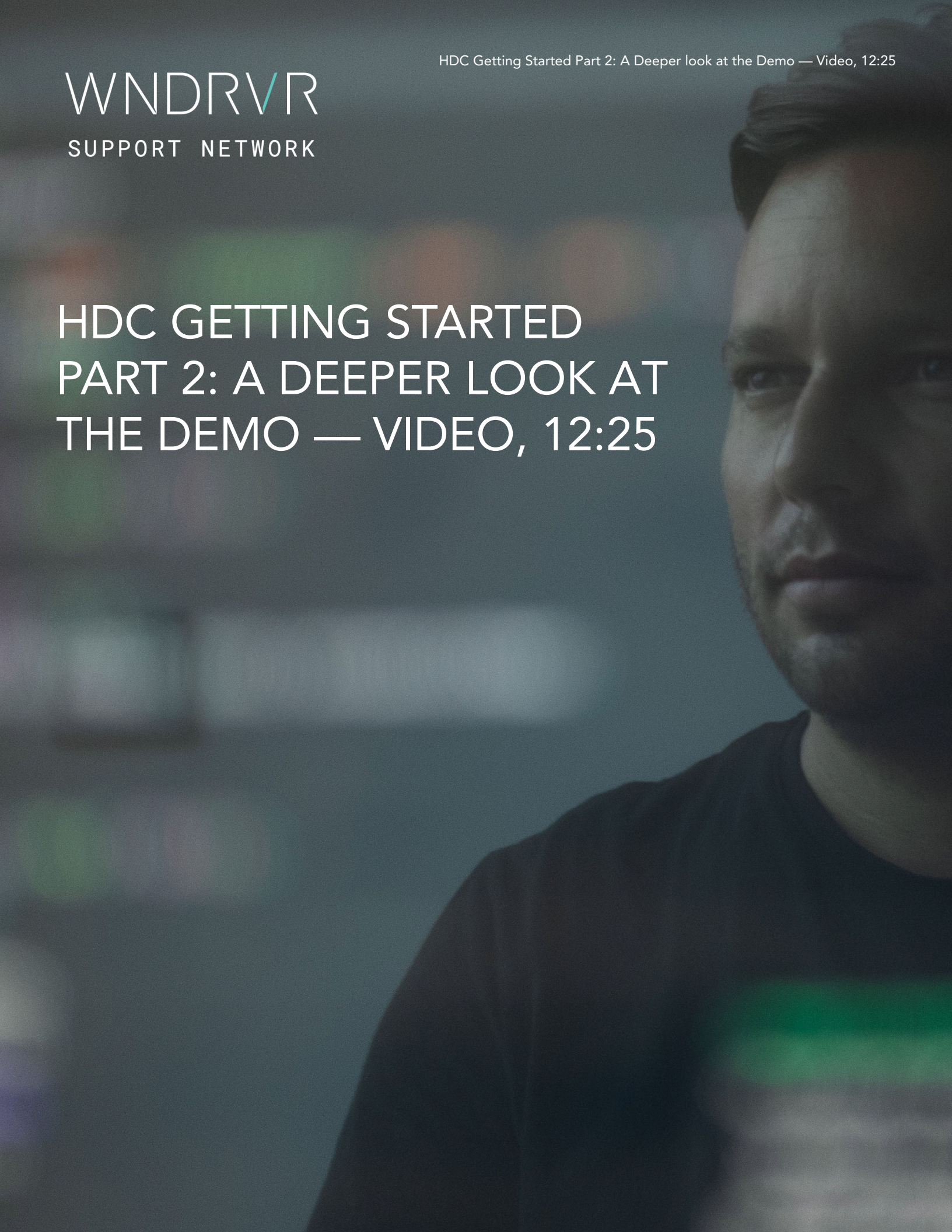
WNDRVR
SUPPORT NETWORK

HDC GETTING STARTED PART 2: A DEEPER LOOK AT THE DEMO — VIDEO, 12:25



Corporate Headquarters

Wind River
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.
Toll free (U.S.A.): +1-800-545-WIND
Telephone: +1-510-748-4100
Facsimile: +1-510-749-2010

For additional contact information, see the Wind River website:

www.windriver.com

For information on how to contact Customer Support, see:

www.windriver.com/support

HDC Getting Started Part 2: A Deeper look at the Demo — Video, 12:25

29 January 2019

1. HDC GETTING STARTED PART 2: A DEEPER LOOK AT THE DEMO — VIDEO, 12:25

Published on 6 October 2015

Transcript

Time (mm:ss)	Narration
From: 00:12	Ok, in the previous session we went through a quick tour of showing how you could bring up a device, and then start sending data to the cloud. I wanted to show that as a very simple exercise, something that could be accomplished in just a couple of minutes, and I provided a number of pre-built pieces in order to enable that. So I did provide a device platform build, an IDP build, running in a VM so that we could just simply use that, before we went through the process of building a platform. I also implemented the data definitions or the data defining in lua scripts, but we'll also show that the basic standard API is a C API, and I'll show you some of the setup that was necessary for that. So, let's take a look now at what I did in order to build this platform
From: 01:03	and to prepare it so that we could do that quick demo. So, let's take a look now down in my "blds" directory, and you'll notice in the blds directory, I have a "p5" directory which is where I actually did the platform build, but over the "cfgs" directory, you'll notice that that's where I keep some scripts so that I can reproduce a platform build, so if I go over and look at that directory, and let's take a look at the scripts that are there, and you'll notice that I have the p5 script, and the main thing to notice about the platform builds, is that you need to include "--with-layer=wr-iot". The wr-iot layer brings in the agent software that runs on the device or on the gateway that communicates with the cloud platform. By enabling this capability on a target device, an IDP target device, then you have that communication up into the cloud,
From: 02:02	and you'll notice that I simply source some of my base information. This is just standard stuff that I typically turn on for all my builds, and saves me the typing in each additional build, but I can add additional builds and I can go back and reproduce a build simply by saving them in scripts. But once you have configured a project, and I'm not going to talk too much about the configure and make of projects, we do have training material on IDP platform and builds with including layers and packages, and all that's covered in some of our standard training material, so I just want to focus entirely on the Helix Device Cloud in this series of training videos. But let's do take a look at the additional script that's necessary, because once you have completed your configure, before you would run make, you do need to also run the "iot-config" script, and if you'll notice here I've
From: 03:05	also saved this as a script that I can run, and so I run the iot-config script. This is the config script that's provided with the development tools, and I've simply provided this script, just to save myself some additional typing as well, but here's how you would define which sandbox you want to connect to, your model type, your serial number, and you can also include like a serial number prefix. There are a number of other variables that you could set, but these are typically the base ones that you would want to set. So, you would run this script, and then after completing running this script, you would simply run "make". And then once your project is built, you'll have output that looks like this, and then you can deploy it to a USB stick, and you can insert that USB stick into your device and boot it up, and this is all, like I said, covered more in our other training courses.
From: 04:04	In the this series, I'm going to assume that you have a device platform built, and that you're able to bring it up. So, now let's take a look though at some of the other pieces that were used so that we could bring up that device and start posting data, and so now if you look down in

	the code directory, you'll see that I have a number of different pieces here but first, let's take a look at some of the "lua_setup" stuff. In the lua_setup, you'll notice that there's a number of different versions of tarballs here, and these are packages that have been prepared that can be deployed to a target, and simply by modifying the things in this directory, I can then make some changes, and then upload those into the platform, and deploy it to my target. Let's take a look down in the "scripts" directory. This is the lua scripts that
From: 05:00	we used in the previous example, and specifically the postdata script, and you can see here it's a very simple script that simply defines a telemetry data type, and you can see that you can simply pass in on the command line a name, a data type, and a value, and then it'll post that data up to the cloud platform, and that's all that's really necessary in order to do that. Now, of course I've done this in a lua script, and the standard API is a C API. So let's go look at how that would appear in a standard C application. And you'll notice that over here, in the "build" directory, you'll see some examples. These are standard examples that you might want to take a look at. Let me bring this up in an editor, and if you look at at the data, some sample code written in C, and this does essentially the same thing that the little lua
From: 06:00	script did as well, but this is defining more data types. There are data types for Booleans, for floating point, doubles, there are integers and strings, and so you can see going through this example code that they're posting various different types, but the API is very similar to what we showed in the lua script. You simply post it, but before you can post it you either set values you can do sets and gets on these values, and then but you create the object first, do the sets and gets, and post, and this is the how simple the API is actually to use, and in this case you could simply go and build these applications, so let's go down and into this data directory, and you can see that I could do. Well, it's already built, but let me and then build it again, and you can see that it builds that quickly, and it'll produce an output binary file called
From: 07:02	simple_tm_test, and then you can run this on your target, just as I ran the lua scripts, and in the previous example and this will register and post data up into the cloud as well. I've provided the makefiles and the sample code right here. In order to do that though, you do need to source the build environment, which I didn't show earlier, but in order to do that, there is an environment script that sets up all the variables, and then you can run this, and I had obviously already done that in this example but, now you can run the make scripts. I showed the example of using a lua script. In order to enable lua, there is simply a module that needs to be provided and that module needs to be installed in the lua path, and so again, building this is just like building any other application. You will run it like
From: 08:00	that, and it will generate a wra shared library, and that needs to be installed in the lua path, and also I've provided another utility which we'll talk more about later. doit and sudoit. These are simply some helper utilities that allow you to run scripts when you invoke them from the cloud platform. We'll talk more about that in another session. Essentially, these are applications as well, just like the other example applications. They simply invoke a shell script, once you have called the application. These would register themselves as well with your base platform. Ok, so once you have compiled and built your applications. Once you have built your platform project, deployed it onto a USB stick, and booted up your project as we did in a virtual machine, or if you're booting up a real gateway device, then you'll be able to connect to it through Ethernet or through Wi-Fi, and then you can actually, you know copy
From: 09:04	down files to it, and then run those scripts. Now of course, application development is typically not going to be running utilities from the command line. You're going to have an application that's either reading a sensor, it's reading some data files, it's reading values out of a database, or its receiving data over some kind of a communication link to a remote device, but that is really more application-specific, and so the examples that we'll show here in the getting started are really trivial, or very simple, just showing how you can read some data. These would simply be put down onto your target, these application components, and then you can use them from the cloud and we'll show that in the next session. Ok, now what I'd like for you to try to do is reproduce these steps. I'd like you to go through and create your platform project, configure it, run the iot_config script to define your variables for your

From: 10:06	connecting to your sandbox, then make the project, and then make the export SDK, install the export SDK, and then go through and make and build the example code that I've shown, and if you can accomplish that and then be able to bring up your device, we've arrived at a point where you're able to to accomplish quite a bit, so let's try this, and then we'll move on to the next section. Ok, so now let's review the steps necessary to reproduce the example. First, you're going to want to create a platform project and you're going to have to do a configure script in order to do that. Here's an example of a configure that will build the project as needed. You'll notice the main thing to include is the layer "wr-iot". Some of these are the things are optional, such as feature remote sessions, but you could also include other packages that you may want to include in your build.
From: 11:04	Then you're going to want to run the iot-config script. This script allows the agent to be configured to establish communication to the server, and you'll specify a server address, a model number, serial number, and any prefix if you, and other options are available as well. You can find that in the the programmer's guide and the user's guide for the IDP platform project. And then after you have configured that, then you're going to want to run "make" and "make export-sdk". This will compile all of the tools and the images and output the tools by installing the SDK. You'll see it and it will install and /opt/windriver if you accept the defaults when you run the script. And then try building some of the sample code. You can go down into the /blds/code/examples directory. Source the environment script that sets up your cross compile environment,
From: 12:04	and then go down into one of the subdirectories and type make and see if you can successfully compile the example code, and then once you have achieved this, we're ready to move on into the next section.

Contact: nlyons

Content ID: 045829

template('WindRiver/function/JS/wrConditionalContent');