B: Example C Client

Prerequisites:

- Installing the Java Runtime Environment
- Installing Ignition
- Installing the following MQTT Modules
 - MQTT Distributor
 v3.1.0 or greater if using Ignition 7.9.X
 - MQTT Engine
 - v3.1.0 or greater if using Ignition 7.9.X
- Downloading the Sparkplug Sample Code onto a development system

Overview:

Sparkplug is an open source project developed by Cirrus Link Solutions which shows how devices or projects can be enabled to communicate with MQTT Engine and Ignition. This example will show how data can be published via MQTT from an emulated device running on a development machine. In addition, it will show how devices or projects can be controlled by writing to tags in Ignition. It will also show the caveats associated with establishing /ending an MQTT session and ensuring that the tag values in Ignition are valid.

Example C Client:

This tutorial assumes:

- Ignition is running and in active trial mode or using a purchased license.
- MQTT Distributor is installed and running, using the default configuration, and in active trial mode or using a purchased license.
- MQTT Engine is installed and running, using the default configuration, and in active trial mode or using a purchased license.

With the standalone Sparkplug example downloaded onto your development machine, change into the directory and build the application. In order for this to work you must have a C compiler installed for your development system. Also, this example assumes the MQTT Server running is MQTT Distributor running with it's default configuration. If you are using a different MQTT Server, edit the following file to reflect your MQTT Server configuration:

sparkplug_b/stand_alone_examples/c/example.c

The most likely candidates for change are the host, username, and password. For simplicity this example does not use or support TLS over MQTT without modifications.

Mosquitto is the only dependency of the sample application. It must be installed on your development machine before building the sample application.

- Mosqutto
 - ° Can be downloaded here. Included in the link are instructions for various platforms. Make sure in the end of installation the
 - development libraries are available on your build path.
 - This is how one would do it in Ubuntu Linux 16.04

 Enter the following commands: sudo apt-get update sudo apt-get install build-essential libssl-dev libc-ares-dev uuid-dev mkdir ~/dev cd ~/dev wget http://mosquitto.org/files/source/mosquitto-1.4.10.tar.gz tar zxvf mosquitto-1.4.10.tar.gz cd mosquitto-1.4.10 make sudo make install sudo ldconfig

With the above steps completed, run the following commands to get the sparkplug library and sample application:

cd ~/dev

git clone https://github.com/Cirrus-Link/Sparkplug.git

cd Sparkplug/client_libraries/c/

make

cd ~/dev/Sparkplug/sparkplug_b/stand_alone_examples/c/udt_example/

At this point the example.c file should be edited to properly reflect your MQTT server URL, port, credentials, etc. Finally, build the application: make

Now simply start the application with the following command: ./example

At this point, the application will start, connect to the MQTT server, publish a Edge Node Birth Certificate, publish a Device Birth Certificate, and begin periodically reporting random data values to Ignition via MQTT Engine. This can be verified via Ignition Designer. Using a Web Browser, browse to the Ignition Gateway on your Ignition Gateway. If it is running on your development machine, that is: http://localhost:8088. You should see this:

i localhost:8088/main/web/home?27	🖾 🖉 R maven download	÷	→☆自				9*	
lg	nition.		admin log out					
Home	Status Configure	Launch Designer	*					
Welc	ome to the Ignition Gateway							
	Congratulations, Ignition is installed and running! Whether you need an OPC-UA server, a SQL datalogger, an HMI, or a full-blown SCADA or MES solution, Ignition can handle it. Here are a few common steps to help get yo Go for the configuration section. The default username and password are: admin / password Chance the password or configuration configuration profile in the Security > Authentication section.	hide this panel						
	Connectively is what ignition is all about. Connect to a PLC on your network using the internal OPC-UA server, or a 3rd party OPC server. Don't have a PLC handy? step or use one of the simulator drivers.	No problem, you can skip this						
	Connect to a database. Database connectivity is at the heart of Ignition's most powerful features, like Transaction Groups and SQLTags Historian. If you have a SQL database, you can gr capabilities by adding a connection to it. No database? You can come back to this step later or skip it entirely.	eatly increase Ignition's						
	Launch the Ignition Designer C ² . This is where the magic happens. Create a project and add windows and transaction groups. Besides the usual status and control functionality, take advantage of resporting capabilities.	advanced charting, tables and						
	2 Launch a client. Or two. Or twenty. Web-launched clients can be launched anywhere on your network from the panel below. With Ignition, you don't have licensing restrictions to limit you.							
	Need more help? Visit the expanded Quick Start guide in the user manual.							
🔺 Laur	ch Projects 🔞							

Near the upper right corner, click 'Launch Designer'. This will open the following window after downloading the .jnlp file and executing it. Note the default username/password is admin/password. Type those into the appropriate fields and click 'Login'.



This will bring you to a new Window where you can select an Ignition Project or create a new one. Create a new project by giving it a name and clicking 'Create New Project'.

	Open/Create Project	
Ignition by inductive automation		
Create New	New Project Set	tup
💱 New Project	Project Name	MyDemoProject 📀
	Project Title	
Open Recent	Authentication Profile	default 💽
📝 Standard Demo Dashboard	Default Database	MySQL
	Default Tags Provider	default 💽
	Project Template	Blank 🔽
	Description	
	Gr	eate New Project
		inductive automation

Now you should be in designer. In the left hand side of the main window is a 'Tag Browser' window. In it, expand 'All Providers -> MQTT Engine -> Edge Nodes -> Sparkplug B Devices -> C Edge Node 1 -> Emulated Device'. You should see the following:

	Test	- Ignition-Wess-MBP - Ignition Designer		
<u>File Edit View Project</u> Component Alignment	Shape <u>T</u> ools <u>H</u> elp			
尾 🏠 \land 🦄 🦒 🗈 🛍 🙆 🐻 🐜 😫 🗉 - 🕨 🕸		표애백금 노크A> 수수수 동북판	표 해 축 … < 1 < 네 1	
Project Browser	a 4	×	OPC Browser 🗗	Ф × 🖡
🖅 🔮 Global		Getting Started	<default></default>	• e –
E- In Project		=	🖭 🕸 Ignition OPC-UA Server	
⊕ Gripts				
- Transaction Groups				⇔
- Windows				√.
Tag Browser	6 F	×		2.*
🚜 📵 🔢 🔹 I 🏷 🔹 😻 📅 🏟 I 🌲 📥				.8°
- 🔄 MQTT Engine		New Main Window	New Popup Window	-
🗄 🔯 Data Types				ø
Edge Nodes				
- Sparkplug B Devices				
🖕 🔄 C Edge Node 1				
Er- 🕲 Emulated Device				
E G input				
↓ Solution Device Metric0	C1tiwTKD^Q9: Str	ng 🗐		
E- S Device Metric1	Boole	an		
e 🔄 output			2	
Device Metric2 Device Metric3	16 II Roofe	t2 an		
I Settle Metres	Custom_Mo	or		
- 🗐 Node Control				
🗄 🛄 Node Info				
H- Node Metric0	hello node Str			
- S Node Metric1	Boole	an		
⊕- Node Metric2	13	12 🗸		
Property Editor		×		
## ≜↓ == ₽\$ ₽\$ % -				
		✓ S	now on Designer Startup	
		•		
		O Welcome ×		
			(-,-)% 206 /	910 mb 國
			,	

You will see the result of MQTT Engine seeing a new edge node and device connect to the MQTT Server and publish an NBIRTH and DBIRTH message. As a result, MQTT Engine created the Ignition Tags shown above. These are also dynamically updated as the values change. You can also write to the outputs after you Enable Device Writes from Ignition. This can be done by putting designer into read/write mode. Do so by clicking this button in the menu to enable read/write mode:



Then you can change any of the values on the outputs here:

□ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □ □		ουιεαι
📃 🛛 🕂 🖶 🦠 Device Metric2	12	Int2
📗 📄 📄 🚊 🖗 Device Metric3		Boolean

You should see the output value change. By changing an output what happens is a DCMD MQTT message is constructed in MQTT Engine and published to the MQTT Server. This message is then sent to the sample client where it is received. In a real scenario the device side code should write the the actual output at this point and read back the value. In that case and in this emulated case, a DDATA message is constructed and published to the MQTT server. It is then received by MQTT Engine and where the state of the tag is updated.

This sample code also includes code that constructs the following message types:

- Sparkplug Templates which are converted to Ignition UDTs by MQTT Engine
- Datasets
- Sparkplug Properties which are converted to Ignition Tag Properties
 - ° In order for these to show up in Ignition they must be known Ignition properties such as engUnit, engLow, engHigh, and Quality,

This example also utilizes aliases to prevent the need to send metric names on any message other than the initial NBIRTH and DBIRTH messages.