Logitech Craft
Craft SDK (Preview)

Introduction

Logitech Craft is a wireless keyboard with a premium typing experience and a versatile input dial that adapts to what you’re making — keeping you focused and in your creative flow.

The creative input dial adapts to the app you're using — giving you instant access to specific functions for the task at hand.

The touch-sensitive control lets you feel your way through your creation — for a more efficient and immersive workflow.

Craft Plugin - Getting Started

To get started on creating your own Craft Plugin, you would need the following components connected and installed.

- Logitech Craft Keyboard

- Logitech Options Software installed from hackathon resource site
- Port availability - Communication between Options and plugins go over on port 10134, it is important to ensure there are no conflicts on this port and it is available for options to use.
- Currently websocket is the only communications protocol supported by the SDK. In the future we plan to extend it to BSD sockets.

Supported platforms & Inbuilt Apps

Currently Logitech Options the following platforms

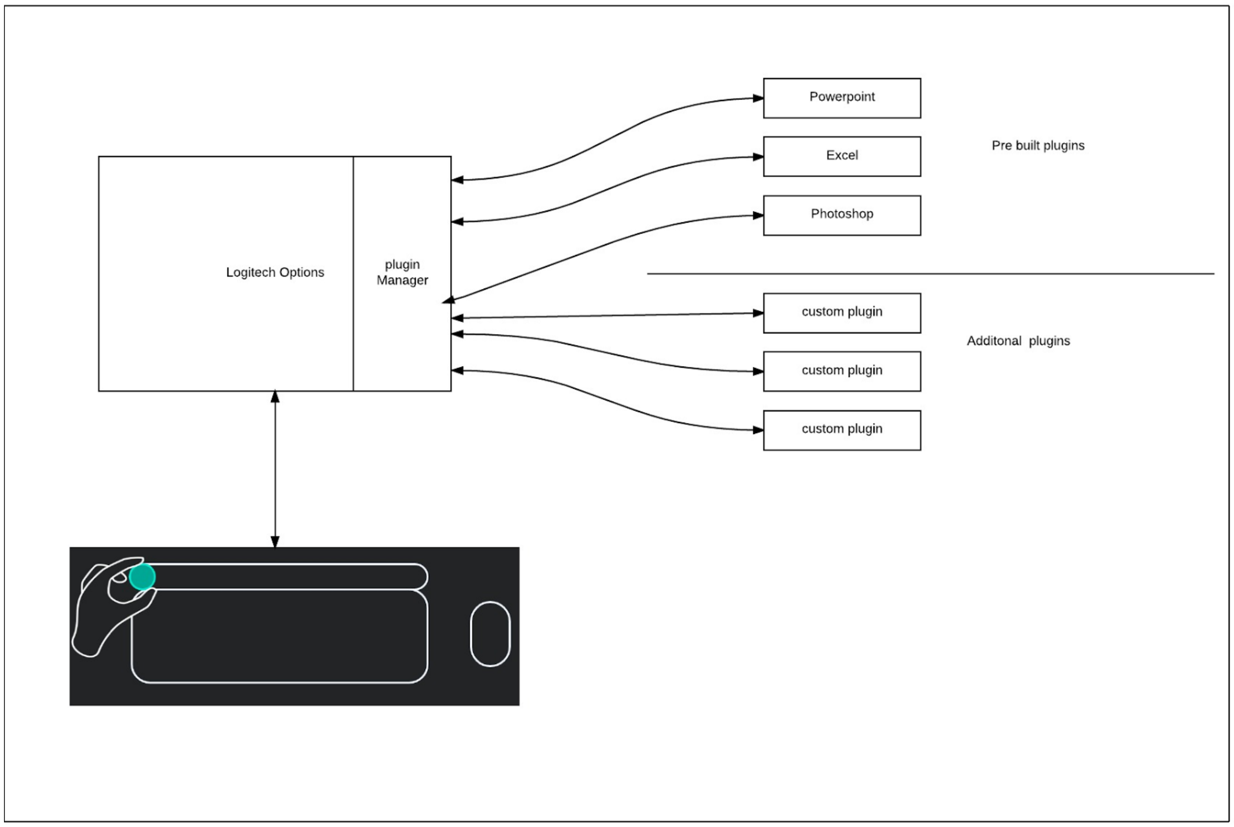
Platform	Version
Microsoft Windows	Windows 7 and above
Mac OS	10.11 and above

In addition, Logitech Options comes with built in support for the following profiles of applications.

Productivity & Creativity

Other Apps

Logi Options & Plugin Overview



The above block diagram shows a high level representation of the the various components and their interactions. At the core of structure is Logitech Options which provides various services that make plugin development very simple and fast. Some services provided by Options include the following

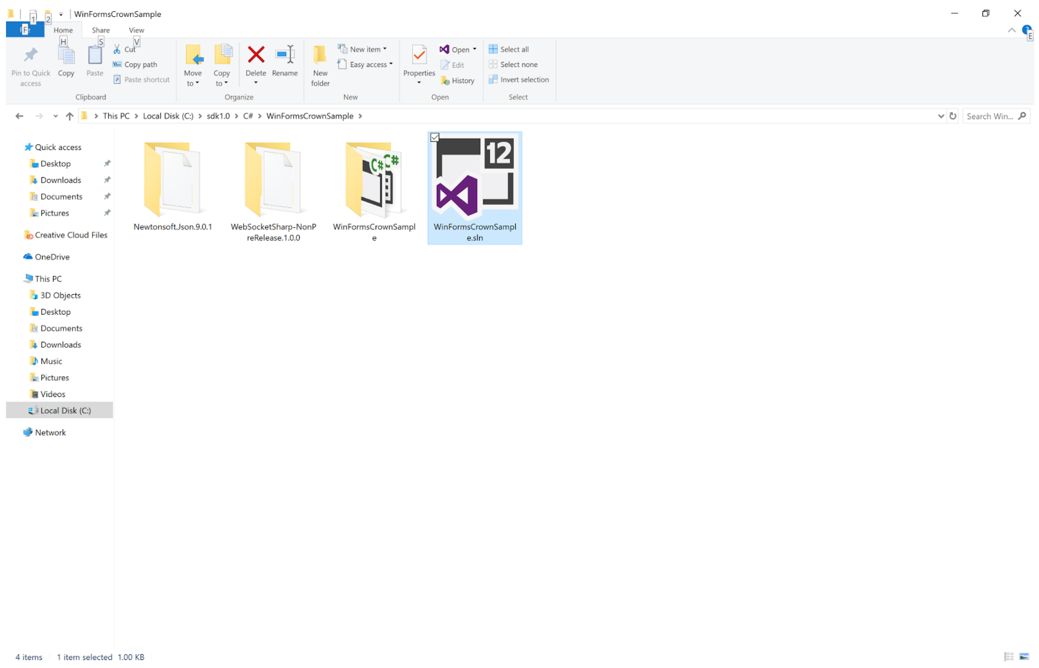
- Hosts a websocket server , facilitating communications with plugins.
- Provides standardized overlay UI elements
- Abstracts all hardware interactions with Craft Keyboard.
- Provides foreground application management to feed the right plugins with crown events.
- Onboarding of plugins from Logitech Options UI

Getting Started - Creating a plugin (C# sample)

We walk through some sample applications that are provided as part of the SDK package. First we examine a simple C# based application, the sample app is located under **sdk1.0\C#** folder.

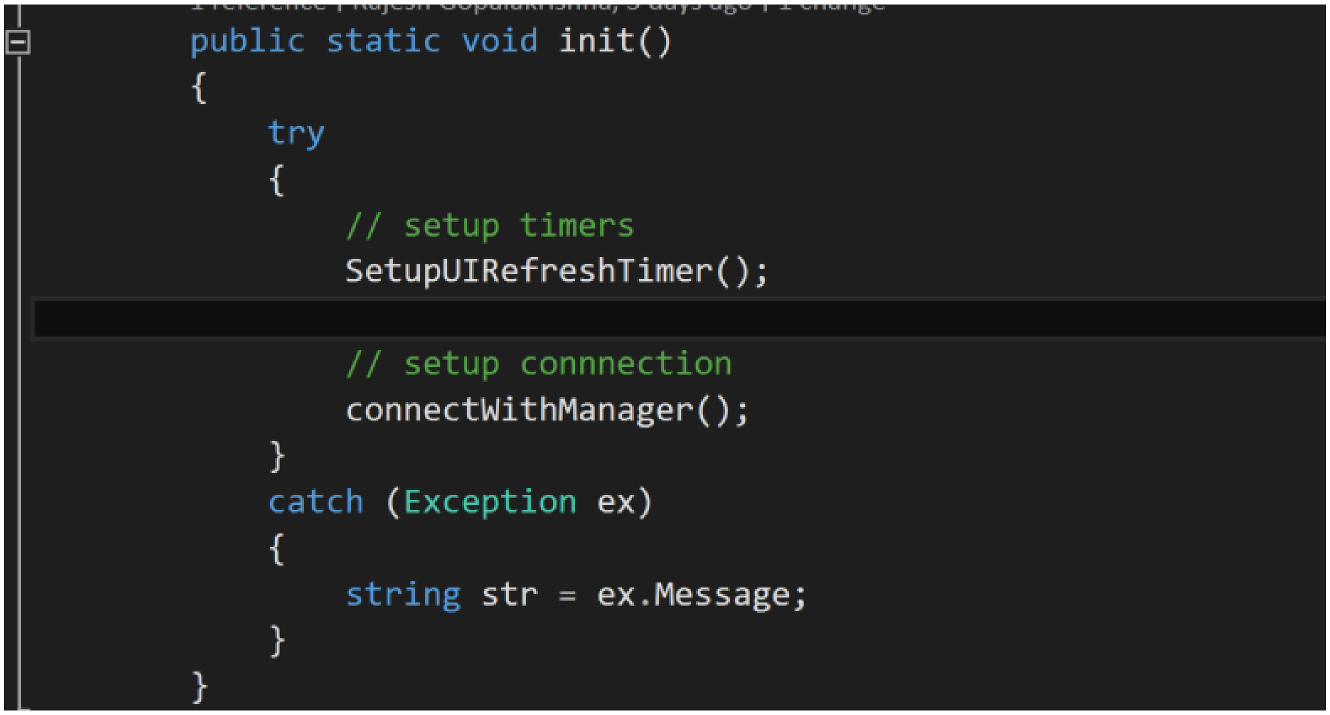
1. Starting the sample app

First open the sample application by double clicking on the WinFormsCrownSample.sln



2. **Initialize routine**

To provide a quick overview, You can open the file Form1.cs , here we examine the function Init()

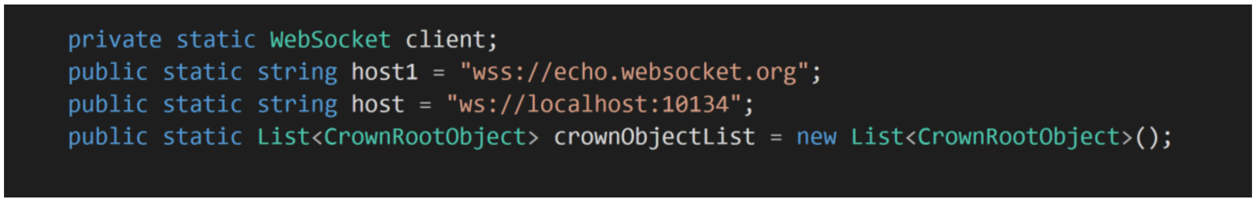


```
public static void init()
{
    try
    {
        // setup timers
        SetupUIRefreshTimer();

        // setup connnection
        connectWithManager();
    }
    catch (Exception ex)
    {
        string str = ex.Message;
    }
}
```

The function provides a connect method to establish a connection with Logi Options and also a timer that is used to demonstrate a simple use case of crown messages.

3. **constants**

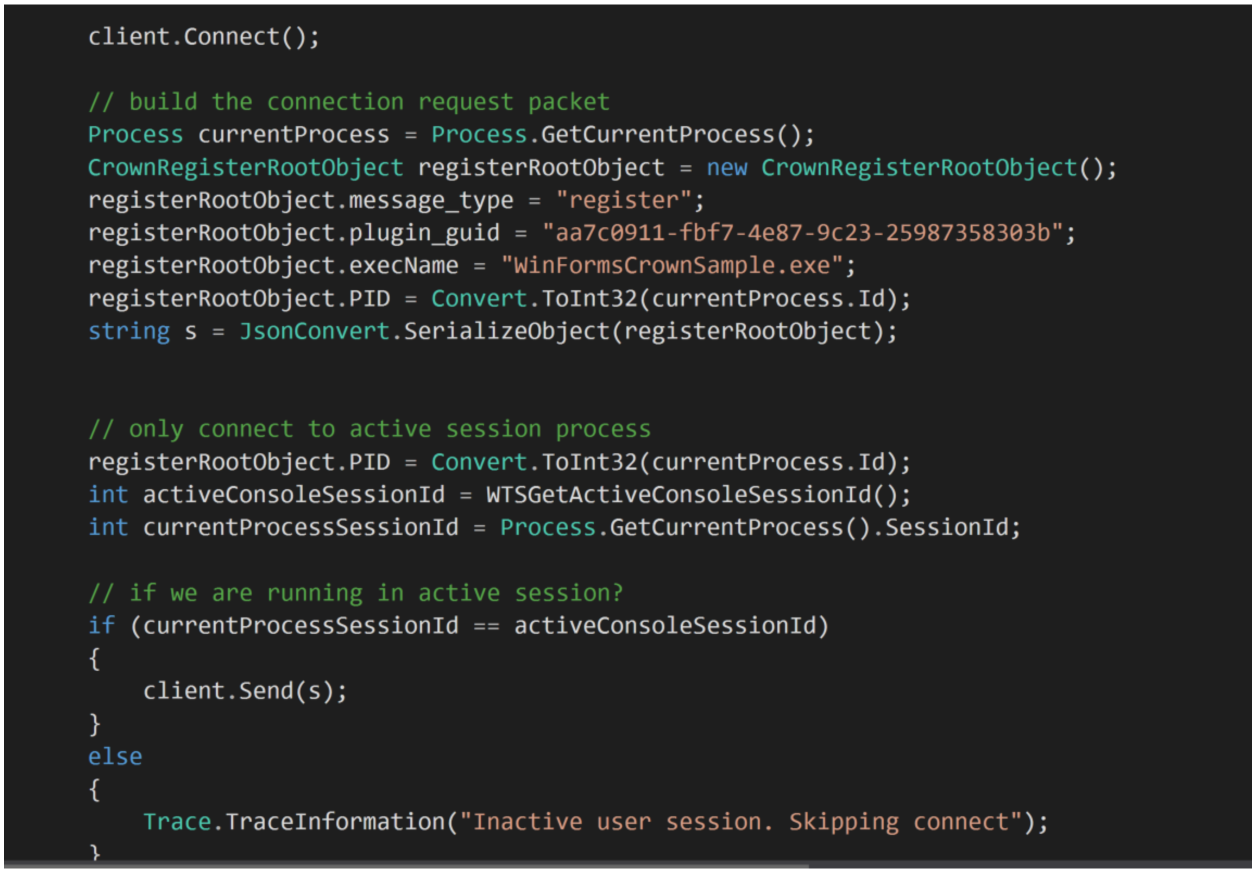
The Connect method employs the standard web socket class to connect to Logi Options



```
private static WebSocket client;
public static string host1 = "wss://echo.websocket.org";
public static string host = "ws://localhost:10134";
public static List<CrownRootObject> crownObjectList = new List<CrownRootObject>();
```

4. **Registering the plugin with plugin manager**

The connect method first connects to Logi Options and then registers itself by providing its guid , pid, exe name as shown in code below



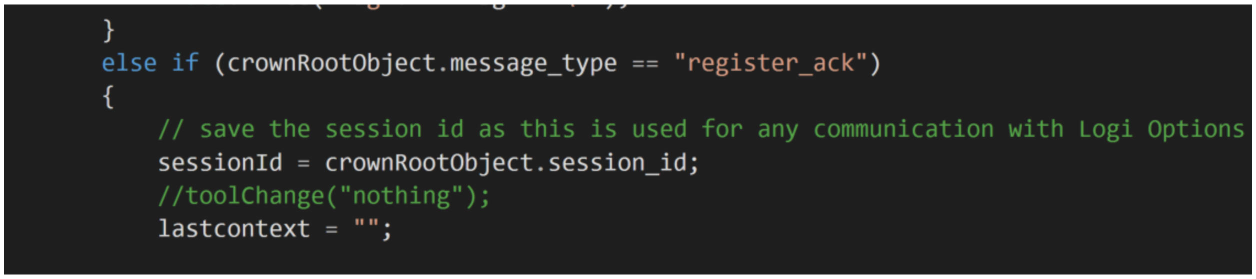
```
client.Connect();

// build the connection request packet
Process currentProcess = Process.GetCurrentProcess();
CrownRegisterRootObject registerRootObject = new CrownRegisterRootObject();
registerRootObject.message_type = "register";
registerRootObject.plugin_guid = "aa7c0911-fbf7-4e87-9c23-25987358303b";
registerRootObject.execName = "WinFormsCrownSample.exe";
registerRootObject.PID = Convert.ToInt32(currentProcess.Id);
string s = JsonConvert.SerializeObject(registerRootObject);


// only connect to active session process
registerRootObject.PID = Convert.ToInt32(currentProcess.Id);
int activeConsoleSessionId = WTSGetActiveConsoleSessionId();
int currentProcessSessionId = Process.GetCurrentProcess().SessionId;

// if we are running in active session?
if (currentProcessSessionId == activeConsoleSessionId)
{
    client.Send(s);
}
else
{
    Trace.TraceInformation("Inactive user session. Skipping connect");
}
```

Corresponding message sent out would be

```
"{\"message_type\":\"register\",\"plugin_guid\":\"aa7c0911-fbf7-4e87-9c23-25987358303b\",\"session_id\":null,\"PID\":12760,\"execName\":\"WinFormsCrownSample.exe\"}"
```

5. **Plugin registration acknowledgement and session id**

Once the plugin connects with Logi Options Mgr and registers itself , Logi Options acknowledges the plugin of the registration with a session id that the plugin needs to use for future communication with Logi Options.



```
        }
        else if (crownRootObject.message_type == "register_ack")
        {
            // save the session id as this is used for any communication with Logi Options
            sessionId = crownRootObject.session_id;
            //toolChange("nothing");
            lastcontext = "";
```

Corresponding json message received would be

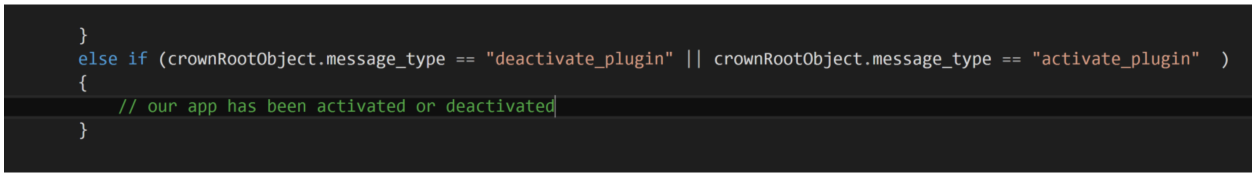
```
{"message_type":"register_ack","sequence_id":0,"session_id":"031e0b9f-2bf6-4576-8f8f-26a5
6d9051a4","status":200,"enable":true}
```

6. **Registration complete and foreground application**

Now that the plugin registration succeeded , Logi Options would forward all the crown events to the plugin whenever the plugin process comes to foreground and is the active process.

7. **Plugin process in foreground and background (activate and deactivate events)**

Logi Options also notifies the plugin when plugin process comes to foreground and background with 2 messages deactivate_plugin and activate_plugin.



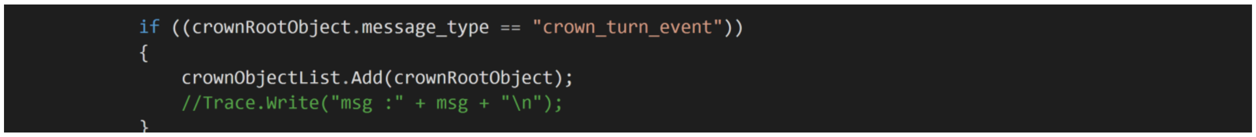
```
        }
        else if (crownRootObject.message_type == "deactivate_plugin" || crownRootObject.message_type == "activate_plugin"  )
        {
            // our app has been activated or deactivated
        }
```

Corresponding json message would be

```
{"message_type":"deactivate_plugin"}
{"message_type":"activate_plugin"}
```

8. **Crown Turn Event**

When an actual crown turn event occurs and assuming our app is in the foreground and is active , we start getting crown turn events .

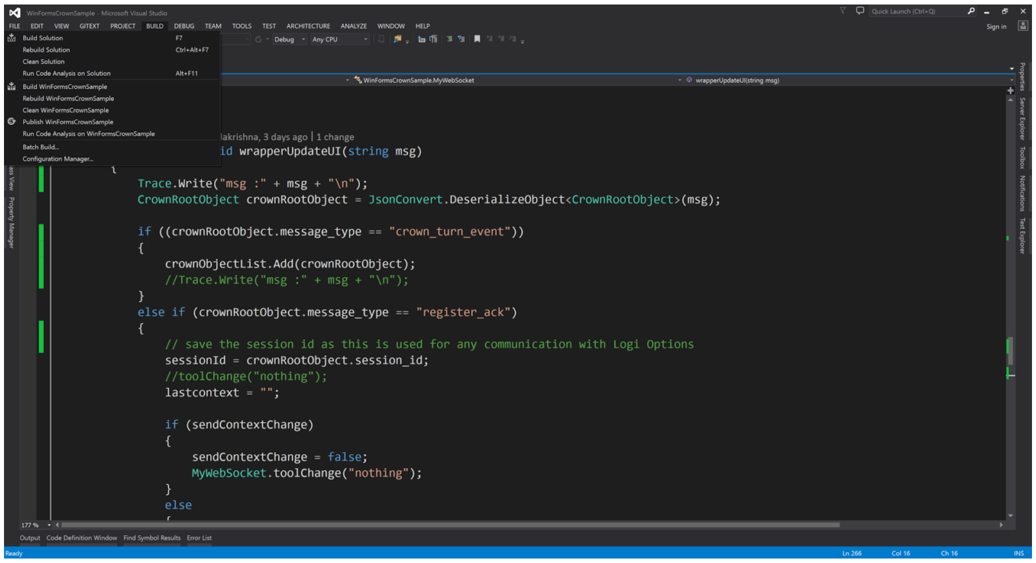


```
            if ((crownRootObject.message_type == "crown_turn_event"))
            {
                crownObjectList.Add(crownRootObject);
                //Trace.Write("msg :" + msg + "\n");
            }
```

{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":3472906}

The crown turn event has 2 fields that we need to pay attention to. The name of these fields are delta and ratchet_delta. Since the crown is capable of reporting fine and high resolution turn events. These get reported in delta field. Since delta field provides granular crown turn events it is ideal for use in cases like brush size .

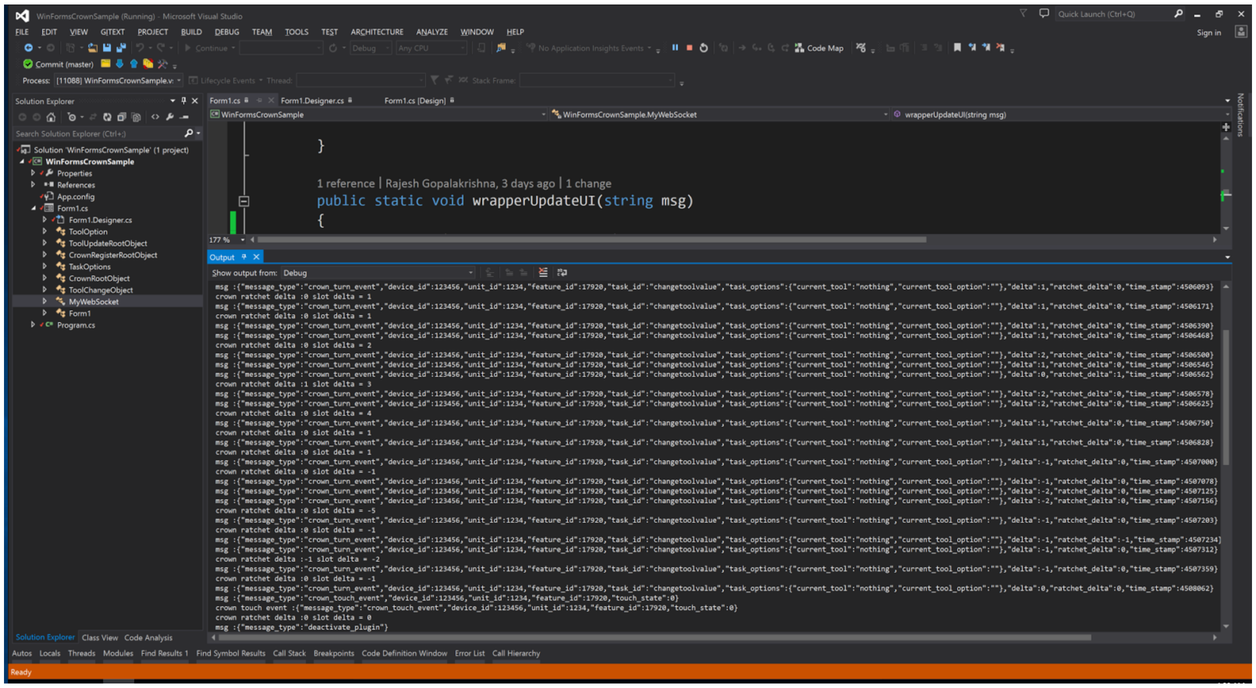
9. Building the WinFormsSample project from VS you can build the project



WinFormsCrownSample - Microsoft Visual Studio
FILE EDIT VIEW GITEXT PROJECT BUILD DEBUG TEAM TOOLS TEST ARCHITECTURE ANALYZE WINDOW HELP
Build Solution F7
Rebuild Solution Ctrl+Alt+F7
Clean Solution
Run Code Analysis on Solution Alt+F11
Build WinFormsCrownSample
Rebuild WinFormsCrownSample
Clean WinFormsCrownSample
Publish WinFormsCrownSample
Run Code Analysis on WinFormsCrownSample
Batch Build...
Configuration Manager...
Debug
Any CPU
WinFormsCrownSample.MyWebSocket
wrapperUpdateUI(string msg)
Output
Code Definition Window
Find Symbol Results
Error List
Ready
Ln 266
Col 16
Ch 16
INS

```
lakrishna, 3 days ago | 1 change
id wrapperUpdateUI(string msg)
        {
            Trace.Write("msg :" + msg + "\n");
            CrownRootObject crownRootObject = JsonConvert.DeserializeObject<CrownRootObject>(msg);

            if ((crownRootObject.message_type == "crown_turn_event"))
            {
                crownObjectList.Add(crownRootObject);
                //Trace.Write("msg :" + msg + "\n");
            }
            else if (crownRootObject.message_type == "register_ack")
            {
                // save the session id as this is used for any communication with Logi Options
                sessionId = crownRootObject.session_id;
                //toolChange("nothing");
                lastcontext = "";

                if (sendContextChange)
                {
                    sendContextChange = false;
                    MyWebSocket.toolChange("nothing");
                }
                else
```

10. You can start debugging the project , notice as you hold the crown and turn the crown , in the debug window you will notice trace messages from the Craft keyboard. We just need to make sure the foreground app is our demo app



```
        }

        1 reference | Rajesh Gopalakrishna, 3 days ago | 1 change
        public static void wrapperUpdateUI(string msg)
        {
```

```
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506093}
crown ratchet delta :0 slot delta = 1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506171}
crown ratchet delta :0 slot delta = 1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506390}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506468}
crown ratchet delta :0 slot delta = 2
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":2,"ratchet_delta":0,"time_stamp":4506500}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506546}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":0,"ratchet_delta":1,"time_stamp":4506562}
crown ratchet delta :1 slot delta = 3
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":2,"ratchet_delta":0,"time_stamp":4506578}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":2,"ratchet_delta":0,"time_stamp":4506625}
crown ratchet delta :0 slot delta = 4
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506750}
crown ratchet delta :0 slot delta = 1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":1,"ratchet_delta":0,"time_stamp":4506828}
crown ratchet delta :0 slot delta = 1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-1,"ratchet_delta":0,"time_stamp":4507000}
crown ratchet delta :0 slot delta = -1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-1,"ratchet_delta":0,"time_stamp":4507078}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-2,"ratchet_delta":0,"time_stamp":4507125}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-2,"ratchet_delta":0,"time_stamp":4507156}
crown ratchet delta :0 slot delta = -5
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-1,"ratchet_delta":0,"time_stamp":4507203}
crown ratchet delta :0 slot delta = -1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-1,"ratchet_delta":-1,"time_stamp":4507234}
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-1,"ratchet_delta":0,"time_stamp":4507312}
crown ratchet delta :-1 slot delta = -2
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":-1,"ratchet_delta":0,"time_stamp":4507359}
crown ratchet delta :0 slot delta = -1
msg :{"message_type":"crown_turn_event","device_id":123456,"unit_id":1234,"feature_id":17920,"task_id":"changetoolvalue","task_options":{"current_tool":"nothing","current_tool_option":""},"delta":0,"ratchet_delta":0,"time_stamp":4508062}
msg :{"message_type":"crown_touch_event","device_id":123456,"unit_id":1234,"feature_id":17920,"touch_state":0}
crown touch event :{"message_type":"crown_touch_event","device_id":123456,"unit_id":1234,"feature_id":17920,"touch_state":0}
crown ratchet delta :0 slot delta = 0
msg :{"message_type":"deactivate_plugin"}
```

Getting started with a plugin (Mac OS) (Objective C)

1. Before you write, build and run a plugin, make sure you have installed the Logitech Options.app in /Applications.

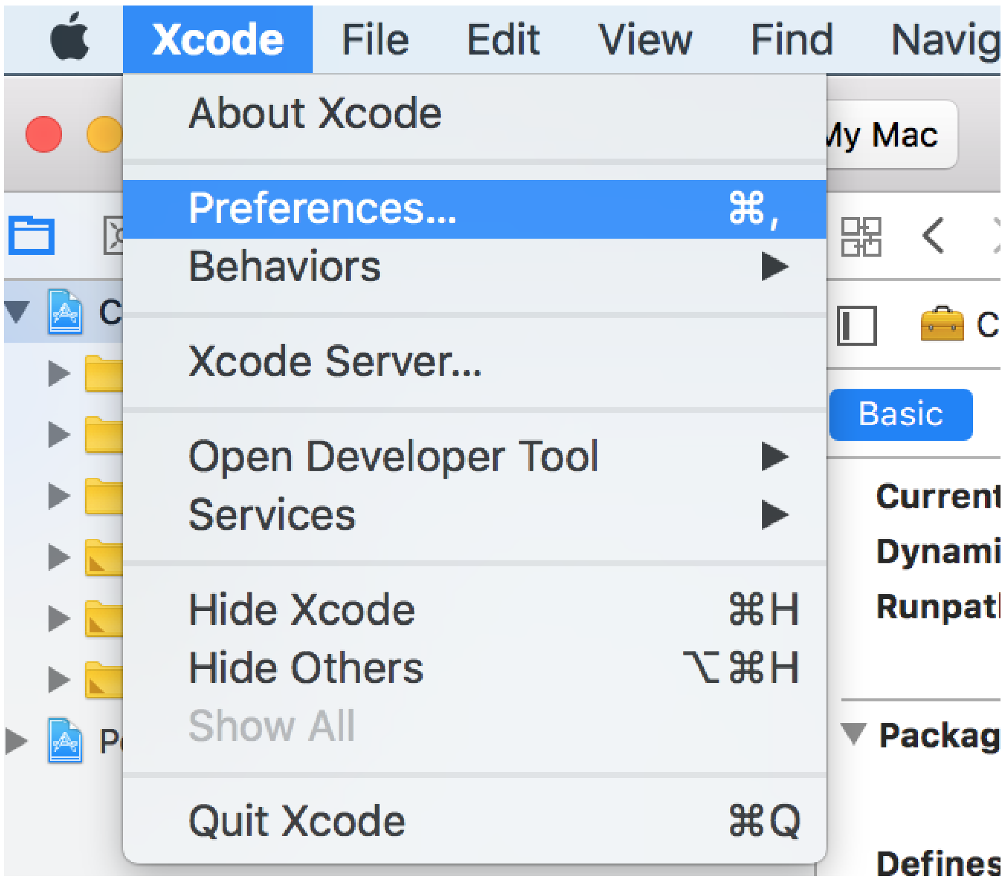
2. Xcode project Craft.xcodeproj builds the Craft sdk framework in Objective-C and some examples that uses this SDK. The project is included in the "craftsdk/Objective-C/examples/Craft" folder.

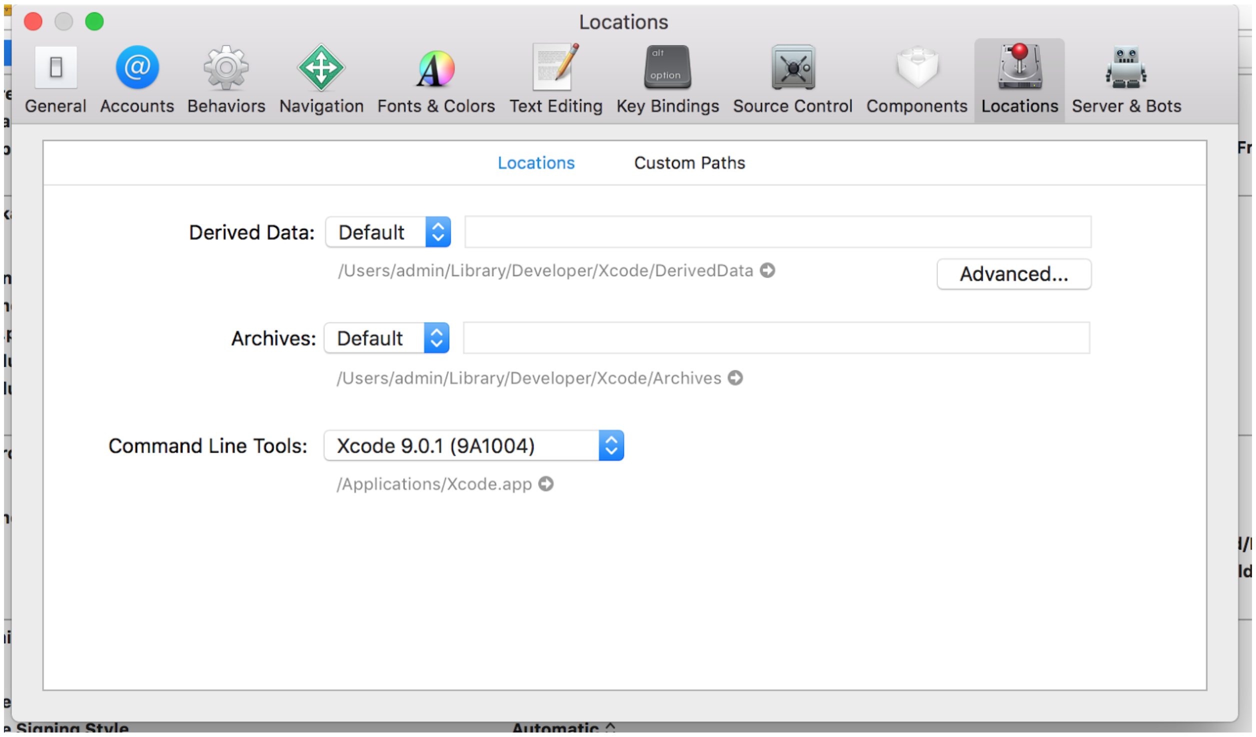
3. The project uses open source websocket protocol implementation called SocketRocket. To include this open source project, the Craft.xcodeproj relies on cocoapods dependency manager. If you don't have cocoapods already installed on your system, you can download it from https://cocoapods.org/

4. In the folder where Craft.xcodeproj is located run command "pod install" and Craft.xcworkspace will be generated

4. Open Craft.xcworkspace with xcode and you will see 3 build targets
- Craft framework : SDK in form of Framework
- craftsdktest : command line sample application that uses above framework
- crafttest : Cocoa UI sample application that uses above framework

5. Use Legacy target settings by going to XCode → Preferences

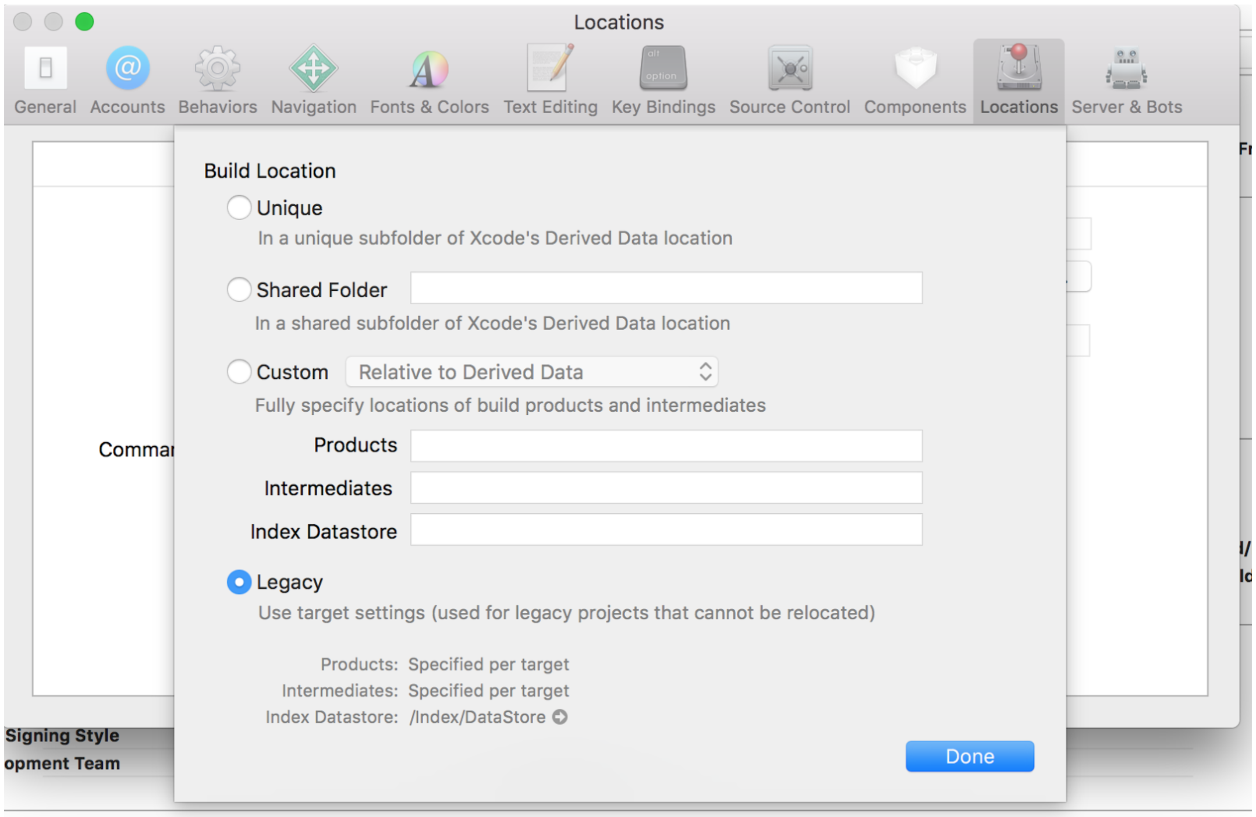



Locations

Derived Data: Default
/Users/admin/Library/Developer/Xcode/DerivedData

Archives: Default
/Users/admin/Library/Developer/Xcode/Archives

Command Line Tools: Xcode 9.0.1 (9A1004)
/Applications/Xcode.app


Locations

Build Location

( ) Unique
In a unique subfolder of Xcode's Derived Data location

( ) Shared Folder
In a shared subfolder of Xcode's Derived Data location

( ) Custom — Relative to Derived Data
Fully specify locations of build products and intermediates
Products
Intermediates
Index Datastore

[✓ Legacy]
Use target settings (used for legacy projects that cannot be relocated)

Products:	Specified per target
Intermediates:	Specified per target
Index Datastore:	/Index/DataStore

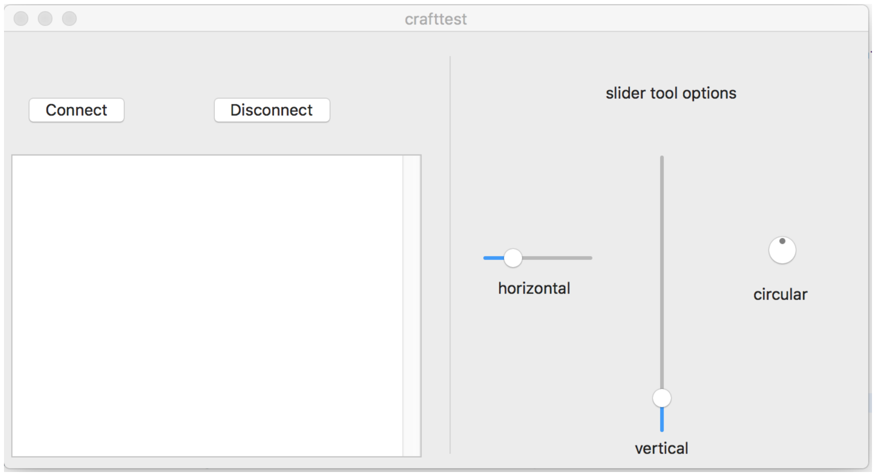
5. The cocoa ui app sample shows how context sensitive overlays can be displayed and

Interacted. The overlay related assets and resources are shipped in the folder "craftsdk/Objective-C/examples/Craft/36f0bda7-7562-4c58-8a25-ddb00e9ebabd"

6. Before building and running the crafttest cocoa UI app,

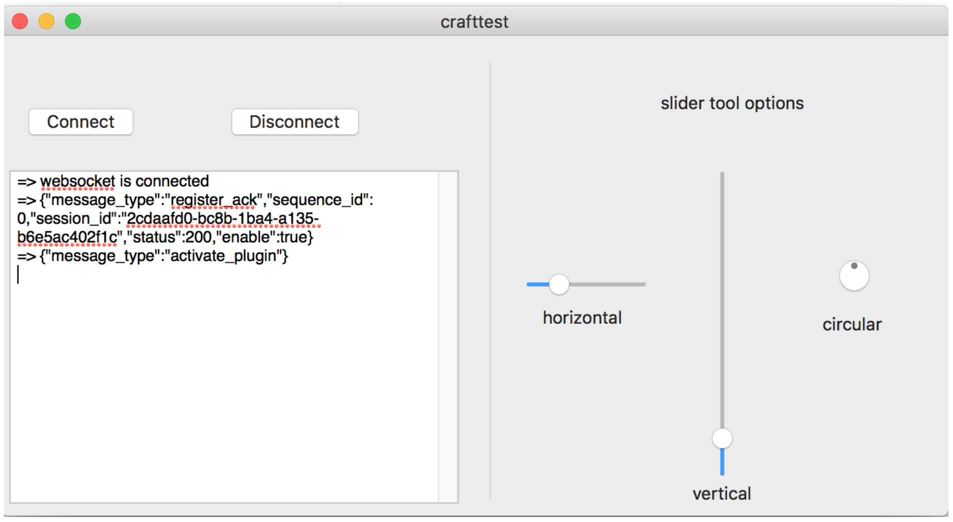
copy craftsdk/Objective-C/examples/Craft/36f0bda7-7562-4c58-8a25-ddb00e9ebabd folder to "/Library/Application Support/Logitech/Logitech\ Options.localized/Plugins"

7. You can now build crafttest UI app build target and when you run you will see



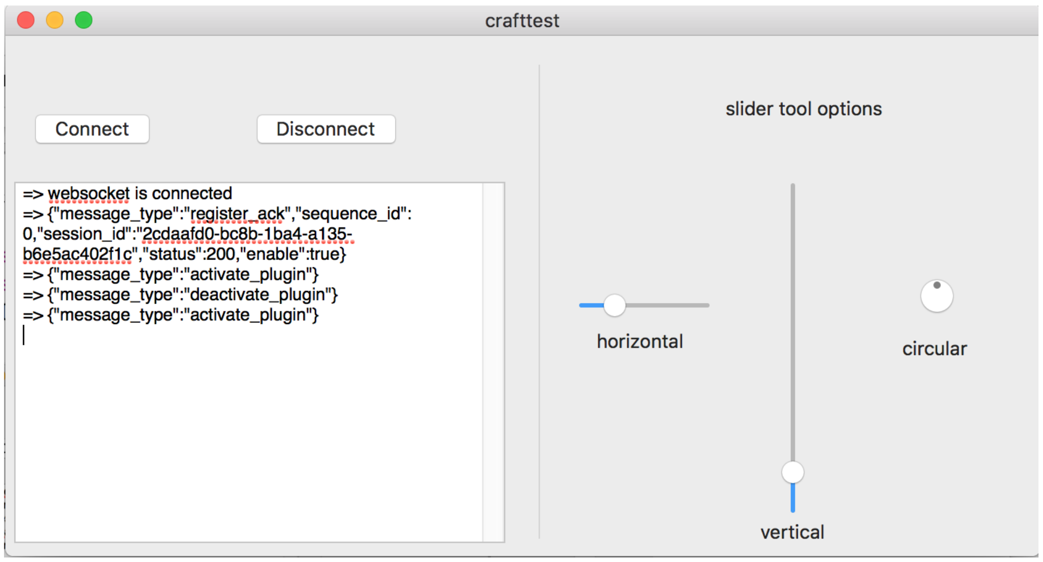
8. Click the connect button to regsiter your app/plugin

```
self.client = [[LogiCraftClient alloc] initWithConnectionType:LOGI_WS];
self.client.delegate = [[BACraftEventHandler alloc] init];
[self.client connectWithUUID:@"36f0bda7-7562-4c58-8a25-ddb00e9ebabd"];
```



```
=> websocket is connected
=> {"message_type":"register_ack","sequence_id":
0,"session_id":"2cdaafd0-bc8b-1ba4-a135-
b6e5ac402f1c","status":200,"enable":true}
=> {"message_type":"activate_plugin"}
```

9. Whenever you app comes to foreground on the Desktop, you will get a activate message from the Logitech's plugin manager and a deactive message when your app window goes into background. You can test it out by cmd+tab 'ing through windows on Desktop



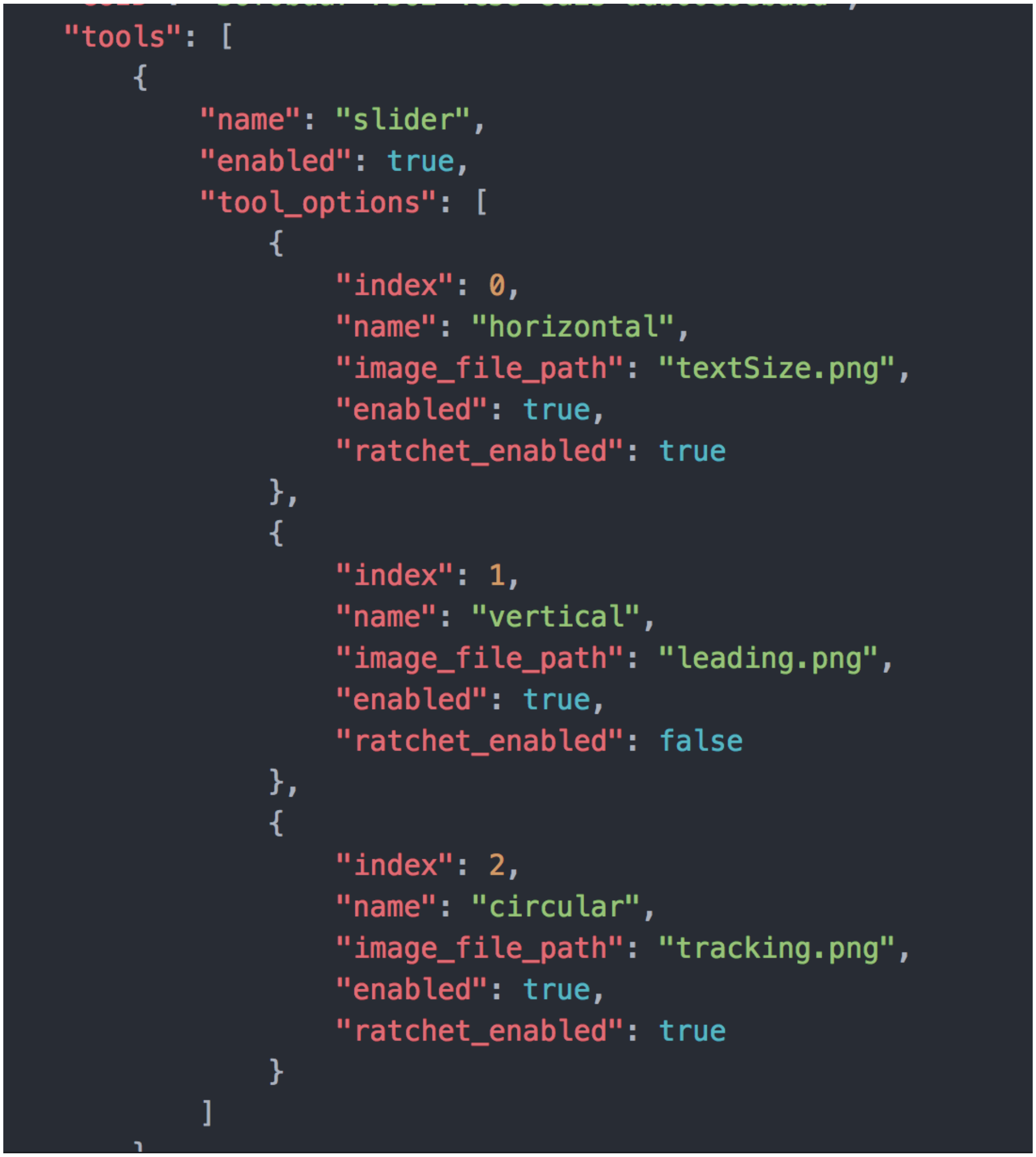
```
=> websocket is connected
=> {"message_type":"register_ack","sequence_id":
0,"session_id":"2cdaafd0-bc8b-1ba4-a135-
b6e5ac402f1c","status":200,"enable":true}
=> {"message_type":"activate_plugin"}
=> {"message_type":"deactivate_plugin"}
=> {"message_type":"activate_plugin"}
```

The context senstive plugins are all about showing the current selected tool , various options present in this tool and letting the user interact with one option at a time. This example has only one tool called "Slider" and has three options under it called "Horizontal", "Vertical" and "Circular". This is captured in the assets and resources folder you copied in step 5.

The handler that is responsible for intercepting the events coming from the Crown and sending updates back Crown plugin manager is

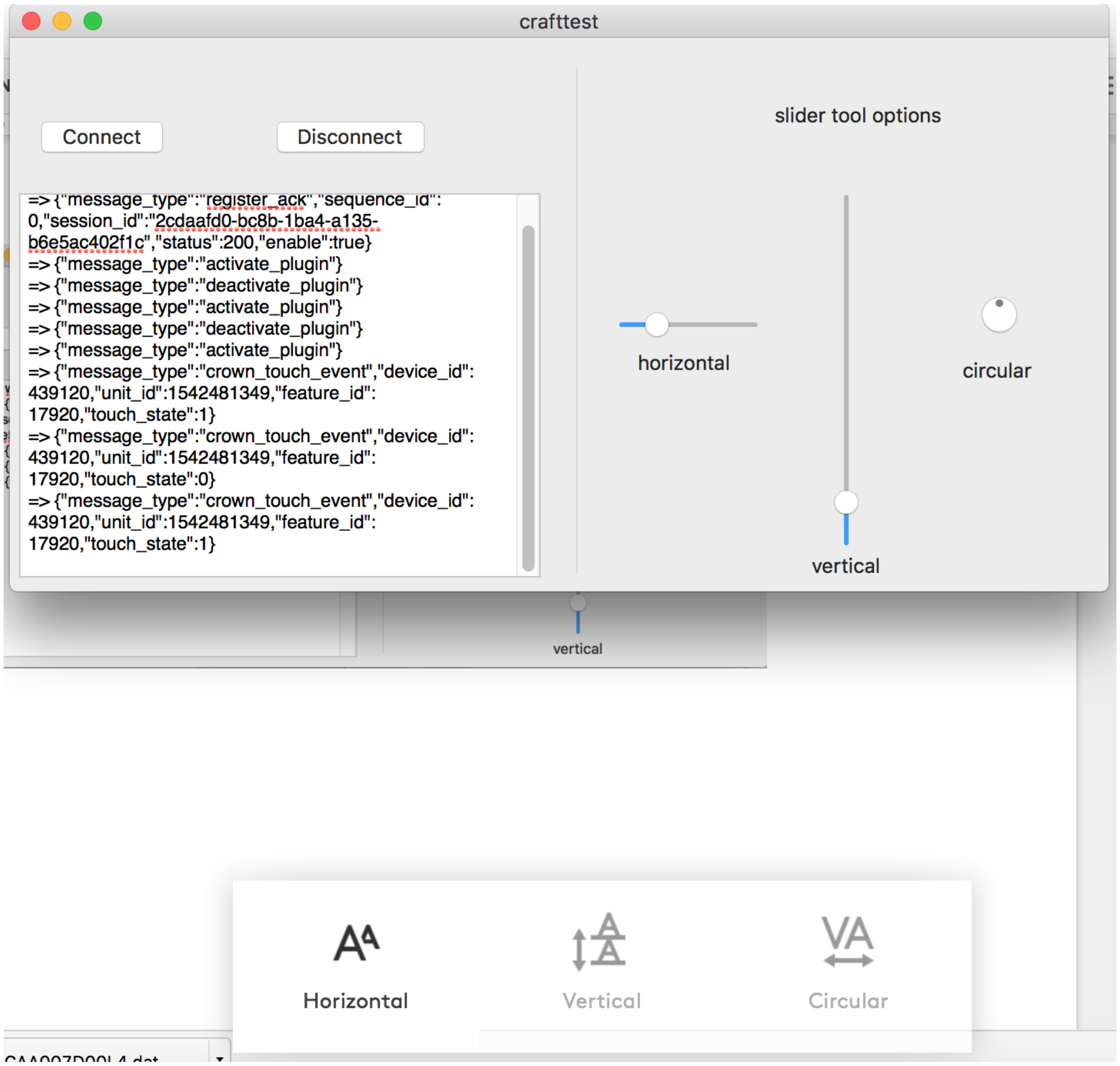
-(void) handleCraftEventViaAppDelegate:(NSDictionary *)msg_dict

The manifest that captures the slider tool , its options is shown below



```
    "tools": [
        {
            "name": "slider",
            "enabled": true,
            "tool_options": [
                {
                    "index": 0,
                    "name": "horizontal",
                    "image_file_path": "textSize.png",
                    "enabled": true,
                    "ratchet_enabled": true
                },
                {
                    "index": 1,
                    "name": "vertical",
                    "image_file_path": "leading.png",
                    "enabled": true,
                    "ratchet_enabled": false
                },
                {
                    "index": 2,
                    "name": "circular",
                    "image_file_path": "tracking.png",
                    "enabled": true,
                    "ratchet_enabled": true
                }
            ]
        ]
```

10. When you touch the crown you should see context sensitive overlays for the Slider tool, each overlay representing one type of slider in the slider tool section

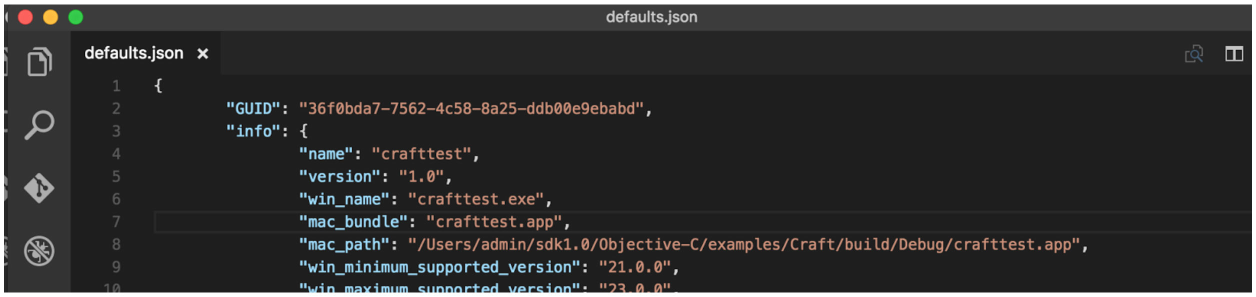


```
=>{"message_type":"register_ack","sequence_id":
0,"session_id":"2cdaafd0-bc8b-1ba4-a135-
b6e5ac402f1c","status":200,"enable":true}
=>{"message_type":"activate_plugin"}
=>{"message_type":"deactivate_plugin"}
=>{"message_type":"activate_plugin"}
=>{"message_type":"deactivate_plugin"}
=>{"message_type":"activate_plugin"}
=>{"message_type":"crown_touch_event","device_id":
439120,"unit_id":1542481349,"feature_id":
17920,"touch_state":1}
=>{"message_type":"crown_touch_event","device_id":
439120,"unit_id":1542481349,"feature_id":
17920,"touch_state":0}
=>{"message_type":"crown_touch_event","device_id":
439120,"unit_id":1542481349,"feature_id":
17920,"touch_state":1}
```

11. You can tap the crown to switch between the overlays to select a particular type of slider option and rotate to update the value of the current selected slider.

12. Making sure the manifest has the right app bundle name and path set

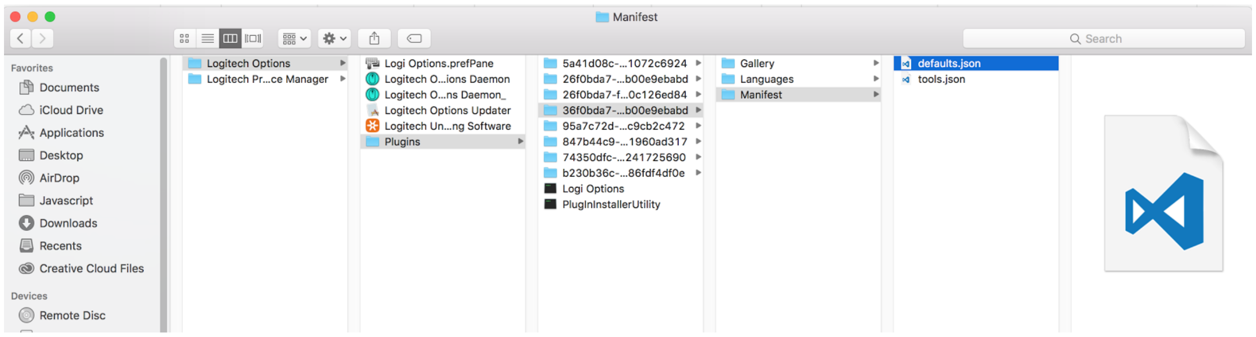
Open your default.manifest file



defaults.json

```
{
        "GUID": "36f0bda7-7562-4c58-8a25-ddb00e9ebabd",
        "info": {
                "name": "crafttest",
                "version": "1.0",
                "win_name": "crafttest.exe",
                "mac_bundle": "crafttest.app",
                "mac_path": "/Users/admin/sdk1.0/Objective-C/examples/Craft/build/Debug/crafttest.app",
                "win_minimum_supported_version": "21.0.0",
                "win_maximum_supported_version": "23.0.0",
```

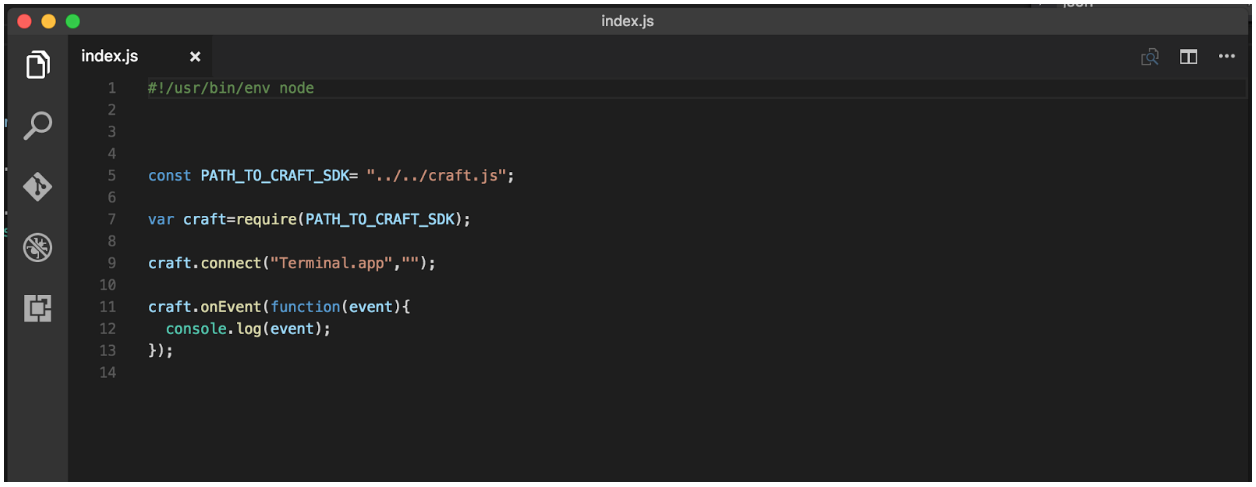
The file is located in /Library/Applications/Logitech Options/Plugins/36f0bda7-7562-4c58-8a25-ddb00e9ebabd/Manifest/default.json



Getting started (Mac OS & Windows) (Javascript sample)

1. First step is to install NodeJS in case you don't have it installed.

2. Next you can browse to simpleExample folder and open index.js for Mac



index.js

```
#!/usr/bin/env node



const PATH_TO_CRAFT_SDK= "../../craft.js";

var craft=require(PATH_TO_CRAFT_SDK);

craft.connect("Terminal.app","");

craft.onEvent(function(event){
  console.log(event);
});
```

3. Let's go through each line of code

```
const PATH_TO_CRAFT_SDK= "../../craft.js";
var craft=require(PATH_TO_CRAFT_SDK);
```

These lines handles all the websocket connection logic which is contained in **craft.js**

4. Next line, handles connection with Logi Options by providing process name

`craft.connect("Terminal.app","");`

In case you are on windows platform change the line to,

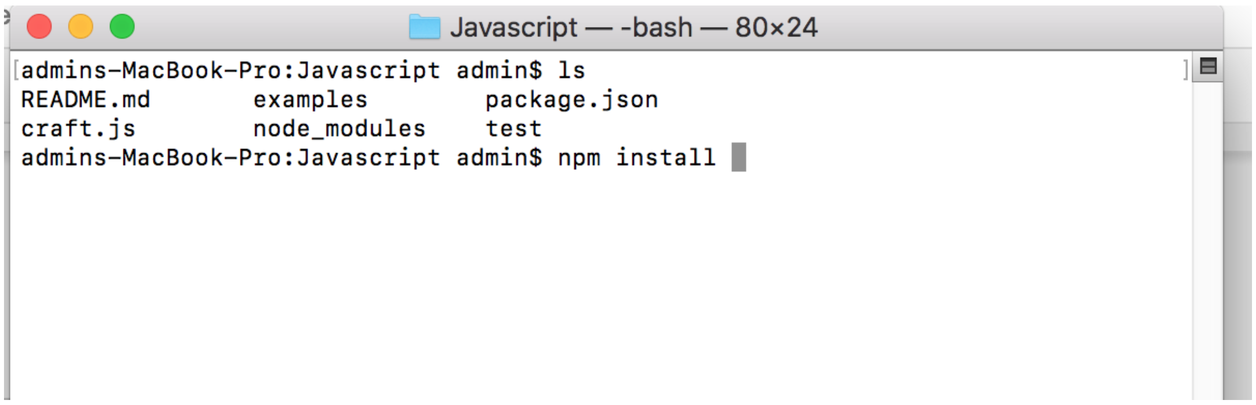
craft.connect("cmd.exe","");

Since on windows we are running under process cmd.exe and on mac it is run under Terminal.app

5. Next we write our event handler to handle crown events

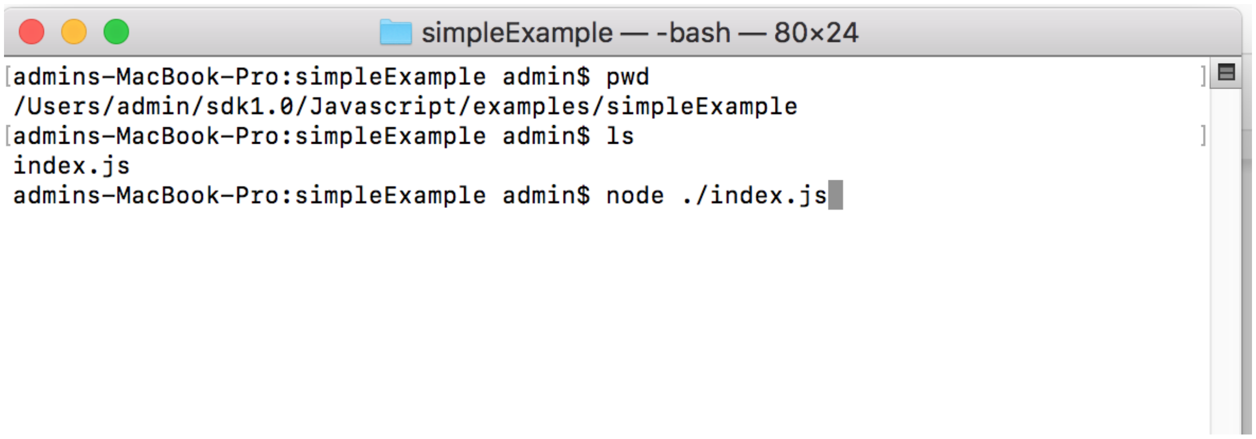
```
craft.onEvent(function(event){
  console.log(event);
});
```

6. You can run sample, after installing all pre req packages by npm install from folder where craft.js is



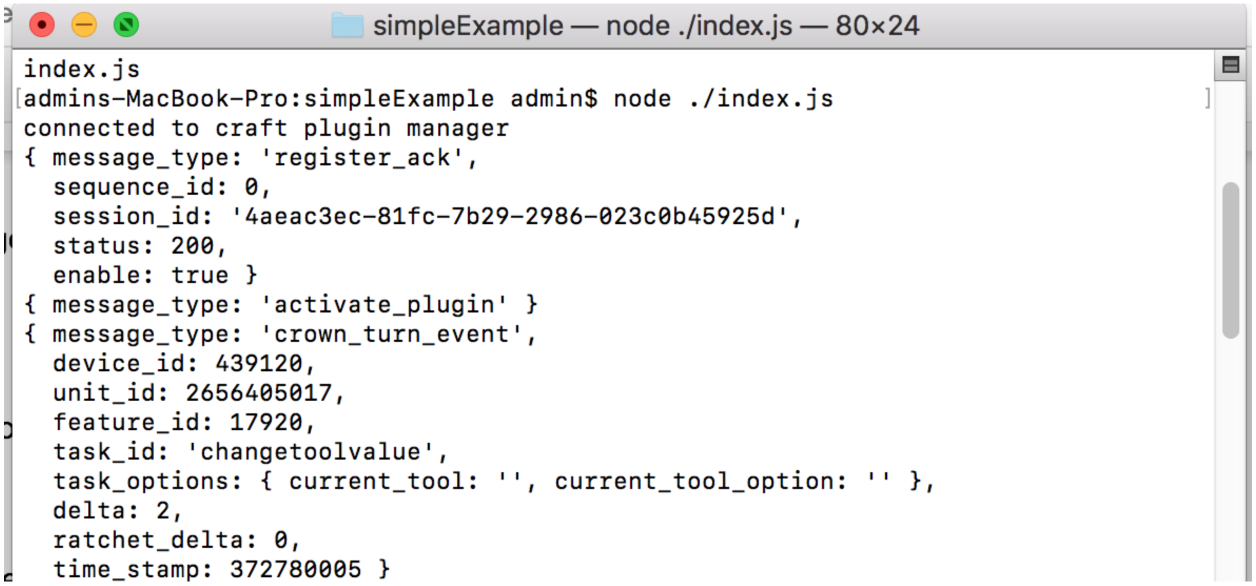
```
admins-MacBook-Pro:Javascript admin$ ls
README.md       examples        package.json
craft.js        node_modules    test
admins-MacBook-Pro:Javascript admin$ npm install
```

7. You can run your sample by executing node index.js



```
admins-MacBook-Pro:simpleExample admin$ pwd
/Users/admin/sdk1.0/Javascript/examples/simpleExample
admins-MacBook-Pro:simpleExample admin$ ls
index.js
admins-MacBook-Pro:simpleExample admin$ node ./index.js
```

8. The output of turning crown is shown below


simpleExample — node ./index.js — 80×24

```
index.js
admins-MacBook-Pro:simpleExample admin$ node ./index.js
connected to craft plugin manager
{ message_type: 'register_ack',
  sequence_id: 0,
  session_id: '4aeac3ec-81fc-7b29-2986-023c0b45925d',
  status: 200,
  enable: true }
{ message_type: 'activate_plugin' }
{ message_type: 'crown_turn_event',
  device_id: 439120,
  unit_id: 2656405017,
  feature_id: 17920,
  task_id: 'changetoolvalue',
  task_options: { current_tool: '', current_tool_option: '' },
  delta: 2,
  ratchet_delta: 0,
  time_stamp: 372780005 }
```

Other Craft Crown Samples

Name	Description	Supported OS
C++ Sample - Breakout	Gaming application	Windows OS
C++ Sample - echo_Client	Console application	Windows OS
C++ Sample - Win32SampleCrownApp	Win 32 application	Windows OS
C# Sample - WinFormsCrownSample	C# based application	Windows OS
Javascript - Nodefighter	JS	both
Javascript - simpleExample	JS	both
Objective C - Craft	Objective C App	Mac OS
Python - Arkanoid	Gaming app	Both
Python - Pong	Gaming	Both
Python - Space Shooter	Gaming	Both

Crown SDK API definition

SDK or API will represent a set of functionalities that will serve as protocol between functional blocks of Plugin Manager, Plugin and Overlay manager.
Plugin manager will implement methods and events that will allow Plugins to connect to Logitech Options and receive events from supported devices.

To ensure compatibility between 3rd party plugin developers and Logitech Options, we will define a set of events and behaviors that every plugin needs to implement. In same way, plugin will be able to talk to Plugin manager when it needs to display an overlay UI. Even plugins developed by Logitech will use the same standard to ensure it is tested and working.

Plugin manager

Plugin manager will listen to the messages on a pre defined WebSocket and handle them according to the messageID and json data included. It will reside in the Options manager as a library. Since we need to allow 3rd party plugin developers to access the crown events, we will define a set of messages that can be exchanged between the Plugin Manager and Plugin. P

Plugin_GUID defined should be unique to every plugin. This id can be set once by developer of the plugin and stored in the manifest file. After plugin registers and connection is established, the Plugin manager will send a new unique_GUID (sssion_id) back to plugin to be used from there on by that specific instance of the plugin, this way the Plugin manager can control which instance of the same plugin will receive the

Register method - plugin registration to the plugin manager

Description -

Register - plugin registers to the plugin manager

- GUID, name of executable, version of plugin, PID (or some way of instance unique)
- We need to make sure plugin runs on different version of application that may have different name of exe.
- If we have a new version of photoshop and old version of plugin we need to notify user to update
- As part of the registration message, plugin will receive a confirmation message including the status

```
{
        "message_type": "register",
        "plugin_guid": "b230b36c-c624-40e3-a120-5486fdf4df0e",
        "PID": “00000”,
        “application_version”: “19.1.0”,
        “application_locale”: “en_US”
}
```

Plugin Registration procedure

1. Open communication protocol
2. Plugin sends GUID
3. If we have a match that means PM already has all the Plugin information, or it is one of the integrated plugins in Options
4. If this is a new GUID then PM requests a whole manifest package from Plugin
5. Once the whole file structure is configured, the PM sends confirmation to the Plugin and a new unique GUID that the plugin will use from there on. This would solve the problem of multiple instances of the same plugin registering with the same GUID.

Tool change event message

This applies to uses cases where there are multiple overlays and we need to switch between the different tools options. The message below can be sent to Plugin Manager to trigger a tool change event.

- Active tool changed on plugin
- “reset_options” - If this option is set to TRUE, the plugin manager will set the current tool option for that tool to the first one instead of continuing from where user left off.

```
{
        "message_type": "tool_change",
        "session_id": "b230b36c-c624-40e3-a120-5486fdf4df0e",
        "tool_id": "brush",
        "reset_options": false
}
```

Plugin

Plugin will be developed as a client model, and will connect to the Plugin Manager open WebSocket. It will also need to handle the messages that the Plugin Manager will send to him.

Messages handled by the plugin (PluginManager send to Plugin)

- Registration status
- Activate
- Deactivate
- Crown event

Registration status

- Acknowledgment of registration
- PM is ready to send events to Plugin
- Status of the registration process (Status codes can be: 200 (registration accepted), 401 (unauthorized))yea
- Session ID is provided which is to be used in all communication between between PM and plugin
- “enabled” field is optional; if present and set to true, plugin operates normally; if present and set to false, plugin is disabled and will not send tool change message upon receiving ACK. Default for “enabled” field is true.

```
{
        "message_type": "register_ack",
        "sequence_id” : 0,
        "session_id": "<randomly generated GUID>",
        "status": 200,
        "enabled" : true
}
```

Sequence_id is a monotonically increasing integer sent with every message for debugging purposes. It will be useful for debugging streams of crown messages. It does not serve any other purpose. The 32 bits unsigned integer will wrap around.

Plugin Activate

- plugin application came to foreground

```
{
      “message_type”:”activate_plugin”
}
```

Deactivate

- Send to plugin that was active

```
{
      “message_type”:”deactivate_plugin”
}
```

-
-

Crown event

- we can have 4 different events (Tap, Press, Turn, Press and turn)
- Device_id and unit_id are type uint, and they represent the unique identification of the keyboard that is sending the event

- Crown Turn
    - State is “start”, “continue”, or “stop”
    - Delta would be the number of steps/degrees reported by the devio
    - Delta negative/positive is turn left or right
    - Time_stamp - 32 bit tick count on Windows, on Mac, 64 bit absolute time in nanoseconds, converted to milliseconds

```
{
        "message_type": "crown_turn_event",
        "device_id": 1111111,
        "unit_id": 1111111,
        "feature_id": 4600,
        "task_id": "changetoolvalue",
        "task_options": {
                "current_tool": "brushTool",
                "current_tool_option": "size"
        },
        "delta": -50,
        “ratchet_delta”: 2,
        “time_stamp” : 34234235,
        “state” : “continue”
}
```