CANON SOLUTIONS AMERICA

radlab
Your Ideas Accelerated
Powered by Canon Solutions America

Research and Development Partnership

Camera Control API (CCAPI)

Download, Setup, and Experience.

Introduction

While testing cameras, lights, software packages, and various lighting and photography techniques used in Photogrammetry, we are testing Canon's new Camera Control API (CCAPI) for its ease of use and for any benefits it may have over the on-board camera controls.

About the API

The camera control API is a RESTful API that will allow users to control compatible Canon cameras via HTTP commands. This allows user to control compatible cameras from almost any OS out there via WiFi.

The first camera to support this API is the PowerShot SX70 HS with firmware version 1.0.1 or above.

Getting the API

Upon opening the box of the camera, I found a note saying that the camera has been modified with firmware to include the (CCAPI) followed by a link. I then visited the link and joined. Once accepted, I began clicking on items that looked like what I was after. This could have been more intuitive.

I found a document on how to connect my computer to the camera (using WiFi) but it wasn't working. I finally went to the video and that cleared up everything. My only gripe was that I had to keep stopping and starting the video so I could perform the steps.

API setup instructions: https://www.youtube.com/watch?v=NwFHUGYzC3Y

Research Notes:

- EDSDK uses PTP (Picture Transfer Protocol) and it's suit for solution development with USB connectivity. On the other hand, CCAPI uses IP (Internet Protocol), and it enables to control camera via Wi-Fi. As CCAPI is OS-independent, Android / iOS / LINUX users who frequently requested the support can utilize this tool for system development.
- Using the CCAPI you can control several aspects of the camera including:
    - -Remotely press the shutter button / take a picture
    - -Control the Lens (Focus and Iris (zoom on select models)
    - -Get and set the camera settings
    - -View directory and file information contained on the memory card
    - -Get Image information (width, height, number of color components , etc.)
    - -Get the live view image on the Electronic View Finder (EVF)

User Experience:

Lauren Williams was the first to toy with the CCAPI. Her background is basic knowledge of Java and Python along with some experience using JavaScript.

"*When I first started working with the API, I had a little bit of difficulty accessing it. I found myself a few times referring to the YouTube video by canon that shows the developer on how to access the API. It was tedious at first but over time I was able to get used to it. It would be easier for the developer to have a more efficient way of accessing the API. I had a difficult time finding what language the CCAPI is compatible with to access it on a mobile device (Android & IOS). It would be nice if there was support for the API. However, I found the CCAPI reference guide helpful when trying to figure out how to access certain features of the API.*"

Next was Lily Haas (high school Summer intern). Her background is basic knowledge of C#, C++, Swift, & Bash. She is familiar with Python and Javascript.

Lily was fortunate enough to get some assistance for a day from Father David Loeffler, who was in town visiting. Turns out Father is quite the tech nerd and loves programming.

From Fr. David

“Working with CCAPI could be considered from either a consumer or developer perspective. From the consumer perspective, the API is not likely to be used unless Canon develops a GUI that would allow users to control the camera without having to know any code. From a developer perspective, the API worked well and had good documentation. I found the thorough listing of error codes to be particularly helpful on a number of occasions.

If I were asked for suggestions, there are two that come to mind. First, it often happened that our first time issuing a “shoot” command through the API produced no result. After that first failed attempt, repeating the “shoot” command without making any changes worked consistently many times in a row. I did not spend too much time trying to figure out why, but there might be a bug to search for.

Second, getting a stream of the camera image was, understandably, a bit complicated. If anything could be done to simplify this procedure, it would be helpful. For most other CCAPI endpoints, simply getting the endpoint yields the desired result. When that did not work for liveview, I was unsure how to proceed. I did eventually find a pdf addressing this question, but it took a bit of searching. I would suggest including that handout in the API reference. Once I finally had instructions, I discovered that viewing an RTP stream would require additional software. That was doable but again, a disorienting break from the pattern that almost every other endpoint uses. Once that software was installed, three steps were needed: save the output of rtpsessiondesc as an SDP file, start the RTP stream using liveview/rtp, open the SDP file using the newly installed software. While there might not be a way to avoid depending on RTP software, I do think that the process could be simplified by: (a) explicitly noting this dependency is the API reference since it is not part of the common pattern that applies to all endpoints and (b) finding a way to automate the three steps described above (save SDP, start RTP, open SDP) when the endpoint is accessed. This may require the API user’s consent since involves saving a file and executing software, nonetheless, having such an option available might be preferable to do each step manually. (If you found a way to do that, I might suggest adding it to the current endpoints rather than replacing the already existing endpoints. That way, a developer who wants granular control over each step can still do it manually.)”