Alexa Helper

A SmartThings Suite of SmartApps for controlling your home with your voice



Written by: Michael Struck

Documentation Version 1.0.0

Contributions:

Code contributions: Tim Slagle, Alex Malikov

Documentation contributions: Sabastian Faust

Licensed under the Apache License, Version 2.0 (the "License"); you may not use Alexa Helper code except in compliance with the License. You may obtain a copy of the License at:

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

All product and company names are trademarks™ or registered® trademarks of their respective holders. Use of them does not imply any affiliation with or endorsement by them.

**What is Alexa Helper:**

The suite of Alexa Helper SmartApps are add-ins to the SmartThings mobile application that allow for additional control over your SmartThings devices using the Amazon Echo. The Amazon Echo (“Alexa” as sometimes referred to) already has control over switches and dimmers within the SmartThings environment; however, it cannot natively control other SmartThings devices such as locks, speakers or thermostats. In addition, there is no way to natively trigger SmartThings modes, routines or security states using the Echo. With the Alexa Helper Suite, almost all items within a SmartThings home can be controlled, along with the ability to control other locations, a function not even the SmartThings infrastructure easily allows.

**Definitions:**

**SmartApp-**Add-in to the SmartThings mobile application that allows for functionality outside of what is provided by default. These SmartApps are not stand alone, but only operate within the SmartThings mobile application. SmartApps are open source and written in a form of the Java programming language called “groovy”.

**Parent/Child SmartApp**-A code structure within the SmartThings development environment where a single instance of a parent SmartApp can have multiple child SmartApps associated with it. Alexa Helper uses this structure; there can only be one instance of the Alexa Helper Parent SmartApp and each scenario created within the parent SmartApp is actually an instance of the child SmartApp. This structure cuts down on repeated code within a single SmartApp.

**Virtual Switch**-A device within the SmartThings environment that appears to be a regular switch. However, the device is “virtual” and not tied to a physical device. There are three types of virtual switches we will discuss when using Alexa Helper: A Simulated Switch, a Momentary Tile and a Virtual dimmer.

**Simulated Switch**-Just like an on/off switch on the wall, when you turn on a simulated switch it remains on, and when you turn it off, it remains off. This type of virtual switch is great when are performing a function that has two states of action. For example, when you go to bed you can say *<“Alexa, turn on Night Mode”>* and a set of functions can occur. When you say *<”Alexa, turn off Night Mode”>* another set of actions can occur with just this one switch.

**Momentary Tile**-A type of virtual switch that behaves like a spring loaded physical switch. In other words, it is in a normally closed (off) state and when pressed it comes on momentarily and the turns itself off. This type of device is good for items that just need an on trigger, such as changing modes or routines.

**Virtual Dimmer**-Another type of virtual switch that acts as a dimmer. You can not only turn this virtual device on or off but you can also set different levels (0 to 100) of “brightness”. Again, this is just a virtual device that is not tied to any physical device in the SmartThings environment. This type of device is appropriate for controlling the volume on a speaker, or setting different temperatures on a thermostat.

**IDE**-Abbreviation for “Integrated Development Environment”. In the context of the SmartThings environment, this is the web site located at <http://ide.smartthings.com> that allows for advanced functionality within the SmartThings environment. This allows you to set up custom devices and applications that appear within your SmartThings mobile application.

**GitHub**-A web-based repository that allows for distribution revision control and source code management functionality.

**OAuth**-Provides client applications a secure method for accessing resources on behalf of the resource owner. Within the context of SmartThings, certain custom applications that allow for remote access (such as remotely turning on or off a light) will require additional authentication. As the owner of the resource, you can authorize remote access to an application by enabling OAuth.

**Access Token**-A security identity that is linked to OAuth and in the context of SmartThings, allows for remote control of a locations devices.

**URL**-Uniform Resource Locator, or basically the address of a web site or online resource, such as <http://www.smartthings.com>

**Conventions:**

The following conventions are used throughout this manual.

**Web sites/URLs** - <http://www.smartthings.com>

**Website links**-Save, Publish

**Spoken Commands** - *<”Alexa, turn on the front porch”>*

**Keyboard Shortcuts:** *CTRL+C, CTR+V*

**Warnings/Notices:**

|  |
| --- |
| **Notice** |
| This is a box you will see to give you notices and warnings on things to watch out for during the usage of these products. |

**SmartThings Mobile Application navigation**: *MarketPlace>>>My Home>>>SmartApps*

**SmartThings Mobile Application screen taps:** <*<About program>>, <<Done>>*

**SmartApps Names:** ***Alexa Helper****,* ***Alexa Helper-Cloud Interface***

**Installation:**

While there are a total of four components of Alexa Helper, only the Parent/Child(scenario) apps are required for basic functionality. If you are wanting to add a virtual dimmer for speaker or thermostat control, you will also install the virtual dimmer code. Finally, if you desire to generate URLs for use in ***Alexa Helper***, you will also need the **Alexa Helper-*Cloud Interface*** code as well.

**Alexa Helper-Parent:**

The code for the main parent app is found of the GitHub site:

<https://github.com/MichaelStruck/SmartThings/blob/master/Other-SmartApps/AlexaHelper/Alexa%20Helper.groovy>

While on this site, find the Raw button and click it. This will bring up a blank page with just the code present. Select all of the code (typically *CTRL+A*) and copy It (Typically *CTRL+C*).

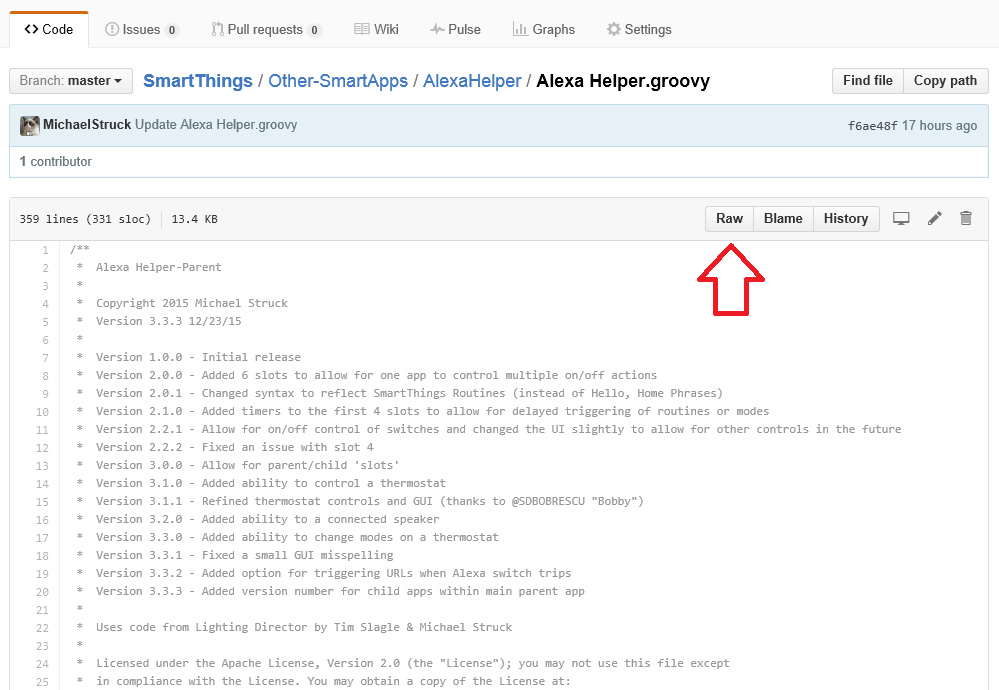


Figure - GitHub "Raw" Button

Next, point your browser to you SmartThings IDE (<http://ide.smartthings.com>) and Log In.

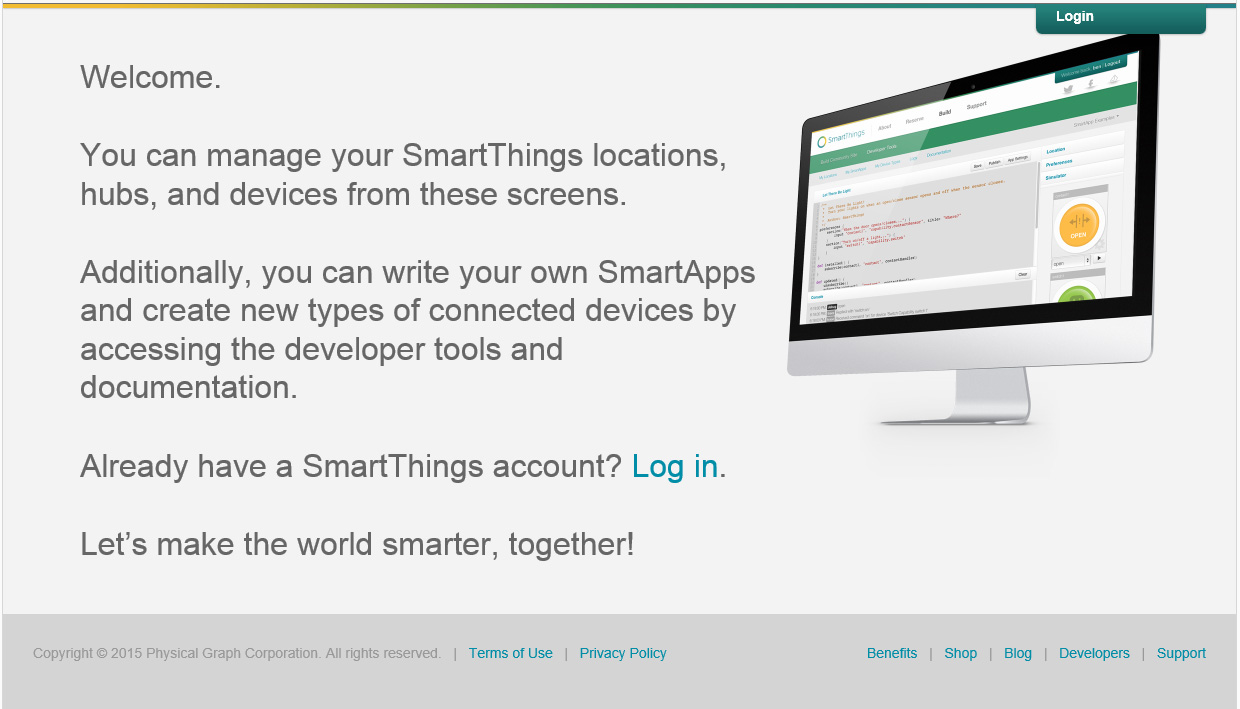


Figure - SmartThings IDE Login

Once you are logged in, find the My SmartApps link on the top of the page.

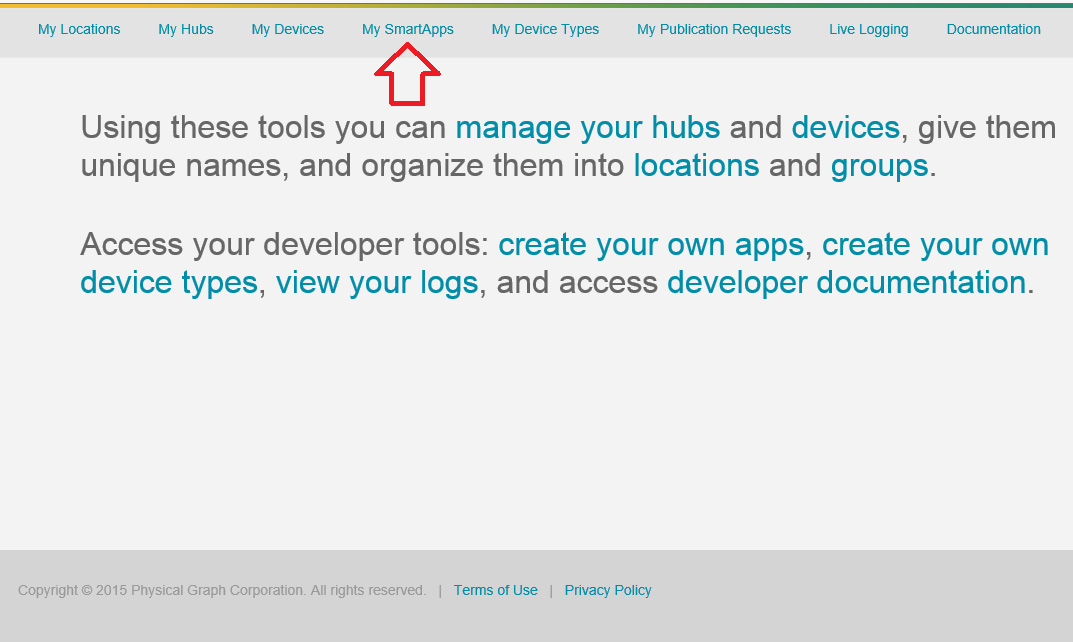


Figure - SmartThings IDE My SmartApps

Clicking this will allow you to produce a new SmartApp. Find the button called +New SmartApp and click it. Since you already have the code in your computer’s clipboard, find the tab along the top section called From Code. In the area provided, paste (typically *CTRL+V*) the code you copied from GitHub. Click Create in the bottom left portion of the page. This will bring up another page, with the code now formatted within the IDE. If the code was copied correctly, there are no other steps except to save and publish the code. In the upper right corner of the page, find Save. Clicking it should give you confirmation the code was saved. Now, click Publish, and you should receive a confirmation of the publishing the code.

Next, you will need to load up the child (scenario app). This code is located here:

<https://github.com/MichaelStruck/SmartThings/blob/master/Other-SmartApps/AlexaHelper/Alexa%20Helper-Scenario.groovy>

The steps for installation are almost exactly the same as above:

* Copy the raw code from the GitHub link above
* Go to the SmartThings IDE page and log in
* Click the My SmartApps link
* Click the +New SmartApp link
* Click the From Code tab
* Paste the GitHub code in the open area
* Click Create from the bottom left corner
* Save the code using the button in the upper right hand corner of the page
* DO NOT publish this code. The child apps do not need to be published as they should only be accessed via the main (parent) app.

If you are an advanced user, you may also want to control two different SmartThings locations using one Echo. This is typically not possible within any SmartThings SmartApp; most apps only control the devices in one location. However, ***Alexa Helper*** can use a URL to send an “HTTP GET” to another SmartThings hub you set up, allowing control of multiple locations from a centralized Echo.

To accomplish this, you will need to install the ***Alexa Helper-Cloud Interface*** code in the remote location you want to control. The code for this SmartApp is located here:

<https://github.com/MichaelStruck/SmartThings/blob/master/Other-SmartApps/AlexaHelper/CloudInterface.groovy>

The steps for installation are similar to the ***Alexa Helper*** Parent Application:

* Copy the raw code from the GitHub link above
* Go to the SmartThings IDE page and log in
* Click the My SmartApps link
* Click the +New SmartApp link
* Click the From Code tab
* Paste the GitHub code in the open area
* Click Create from the bottom left corner
* Save the code using the button in the upper right hand corner of the page
* Publish the code using the button in the upper right hand corner of the page

Before leaving this page, you will need to ensure you enable OAuth for this particular SmartApp. This OAuth allows for secure remote access of the SmartThings devices. To enable OAuth, first find the   
App Settings button in the upper right corner of the page.

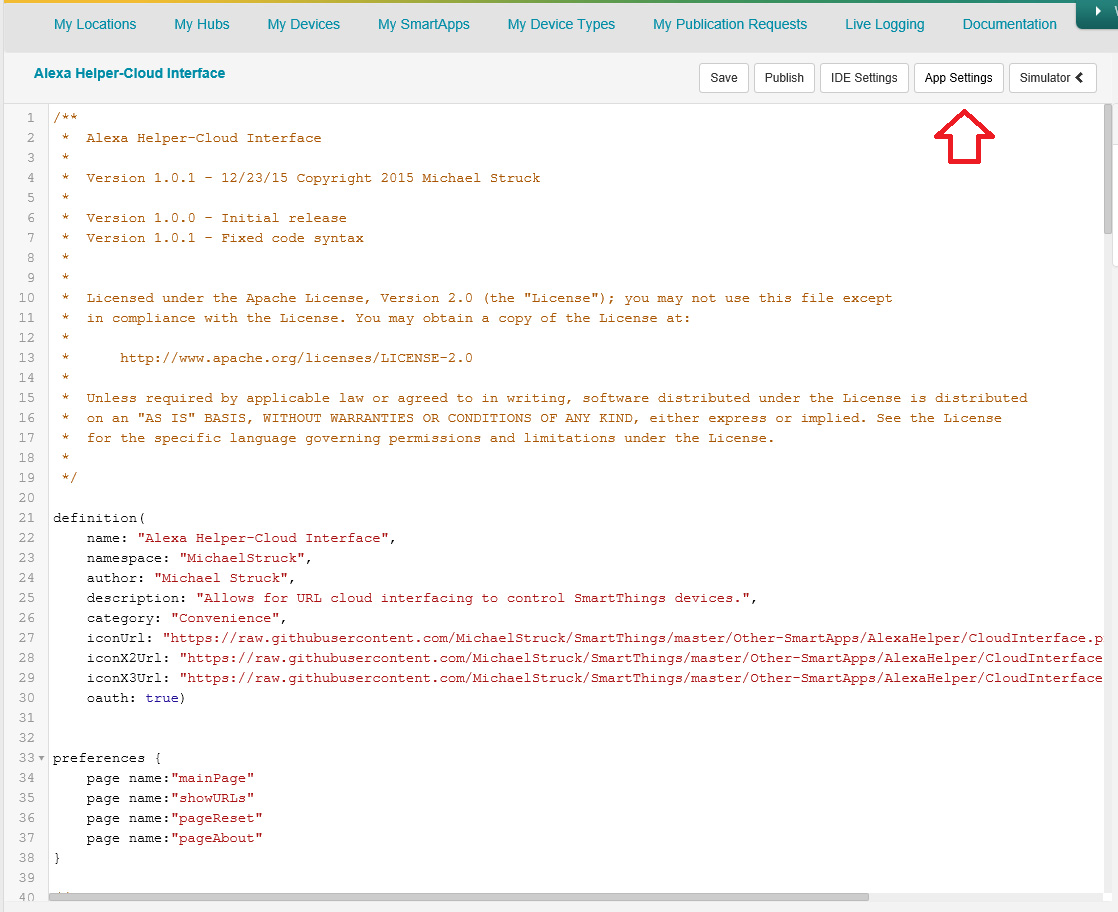


Figure -SmartThings IDE App Settings

From here, find the OAuth section toward the bottom of the page.

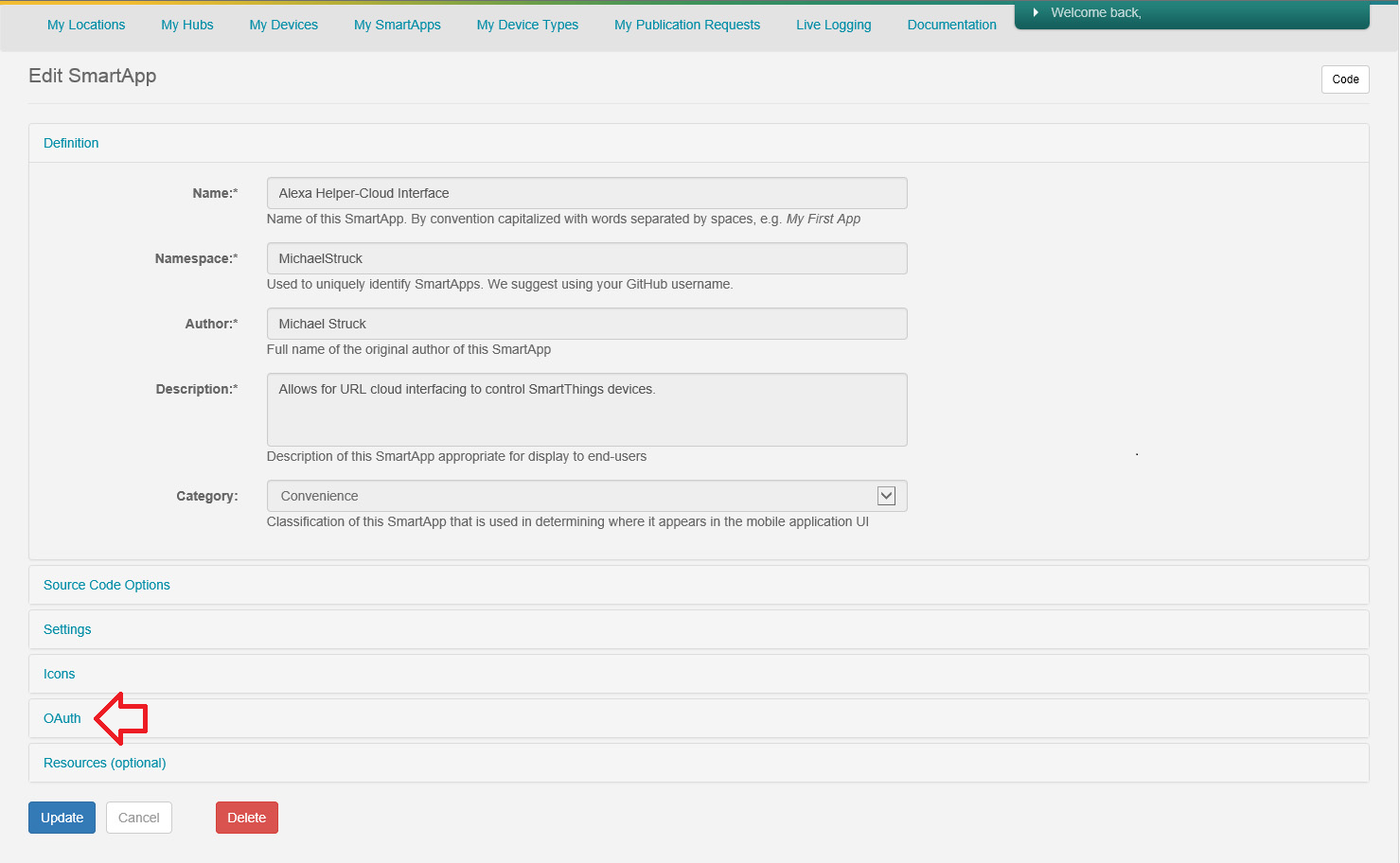


Figure - SmartThings IDE OAuth Settings

Clicking the OAuth link will reveal a button labeled Enable OAuth in Smart App. Click this button. The screen will change, giving you a unique code for your Client ID and Client Secret. These are the foundations of the security of your app and should be kept secret. You will not need to memorize or write down these codes or add any other information to this page; OAuth simply needs to be enabled for ***Alexa Helper-Cloud Interface*** to generate your unique URLs.

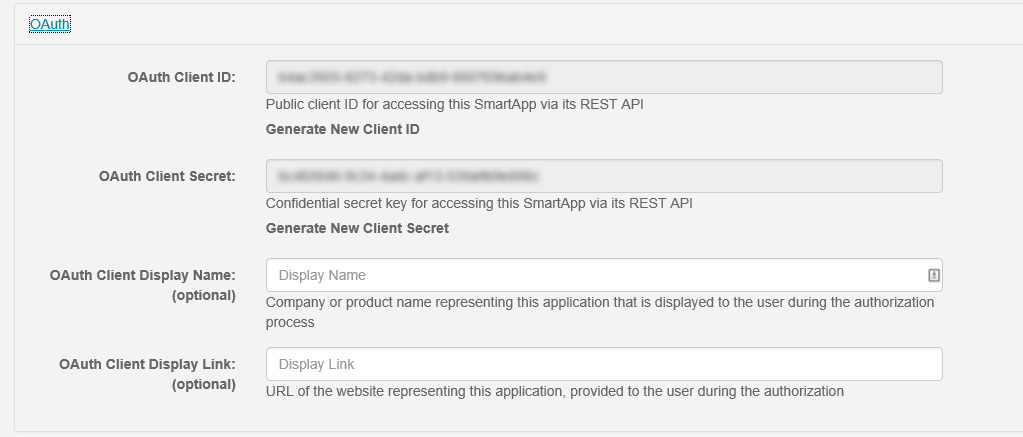


Figure - SmartThings ID OAuth Settings

The final step is to go back to your code using the button in the top left of the page and Save, then Publish the SmartApp again.

**Set up of your Amazon Echo:**

The setup of your Amazon Echo within the SmartThings environment is outside the scope of this document. To use ***Alexa Helper***, it is assumed you have done the following:

* Configured your Echo on your home’s Wifi and that it is functional
* Configured the Echo to work with your local SmartThings devices and ensure you can turn on and off the switches associated with your account

If any of the conditions above are not true, the Alexa Helper SmartApps will not function in your environment. Please refer to your Amazon Echo documentation, use the ***Amazon Echo*** mobile app, or visit <http://alexa.amazon.com> to set up your device.

Once you have a functioning Echo, and you have confirmed it can control your SmartThings devices, it is recommended to set up some virtual tiles and switches to be used by ***Alexa Helper***.

**Virtual Switch Setup:**

While not required, virtual switches allow for better control of your SmartThings environment than using the physical switches in your house. This allows a non-physical switch to be “tripped” to make other events happens, some of which the Amazon Echo cannot perform natively. This includes changing security modes, or running SmartThings routines, which can include a lot of functions that need to happen at the same time. For example, having a virtual switch named “Night Mode” can be tied to a “Good Night” Routine, which could shut off the lights, lock the door, turn off the heat, set the mode to “Asleep” and set the Smart Home Monitor to “Armed:Stay”. All with saying *<”Alexa, turn on Night Mode”>*

Virtual switches are not present by default in your SmartThings Mobile Application; you must gain access to the SmartThings Developer IDE to create these switches. However, the process is rather simple.

Just like adding the code for the Alexa Helper SmartApp, go to the IDE at <http://ide.smartthings.com>.

When you have logged into the site, find the section labeled My Devices. Click this link.

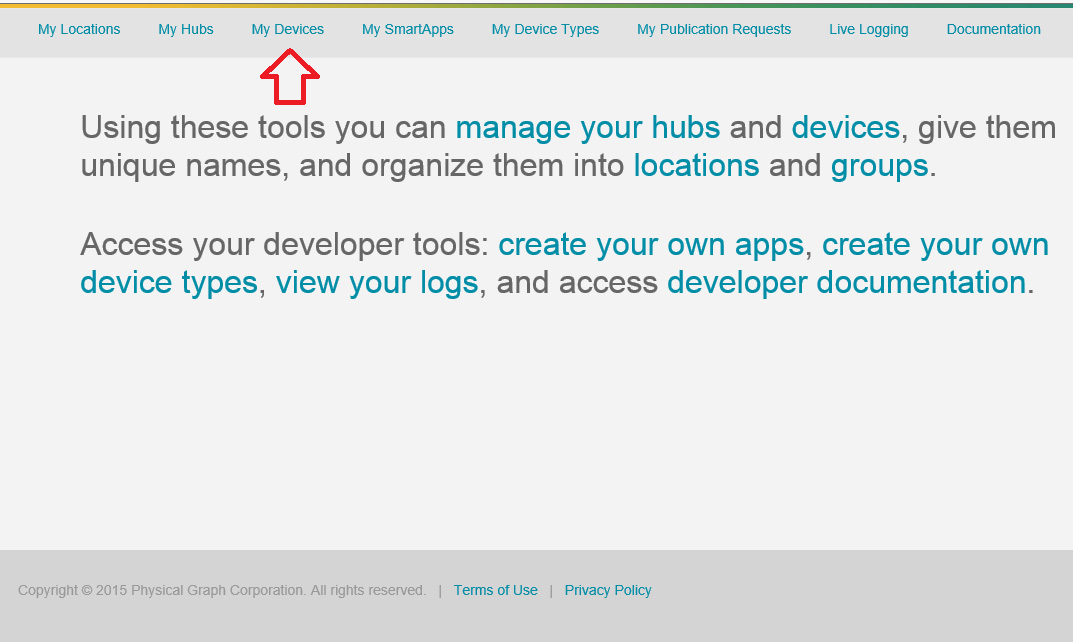


Figure - SmartThings IDE My Devices

On this next page you will be presented a list of all of the devices within your SmartThings account. These are typically the physical devices within your account. From here, find the button in the upper right corner of the page labeled +New Device. Click this link and you will be presented a number of fields to fill in.

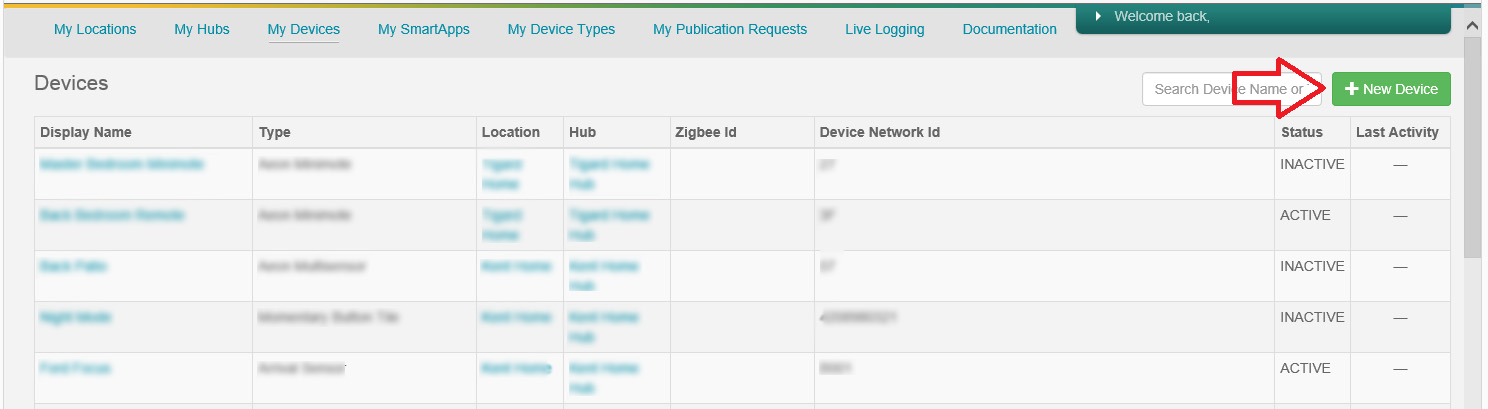


Figure - SmartThings IDE New Device

|  |  |
| --- | --- |
| Starting at the top of the fields, fill out the name of the device. You may want to create a sort of naming convention with starting everything with a ‘v’ such as vThermostat. However, if you plan to use these names with Alexa you should use a name that Alexa can recognize (Like “Night Mode”, “Day Mode”, etc.). You can always change this name later.  Once you fill out the name, simply use the same name used above for the label.  Since this is a virtual device, leave the Zigbee Id field blank.  For the field labeled Device Network Id, fill in a number that is unique to your environment (you may need to look at the list from the previous screen to determine what is unique.) This field is required, however, the number is not used anywhere else, so a simple “1234567890” could work well for this. Any subsequent devices created could build off of this (for example: “223456790”).  In the next field labeled Type, choose either a “Momentary Button Tile” for a momentary switch, or “Simulated Switch” that has two unique states (on/off). | C:\Users\Michael\Desktop\ClipArt-AlexaHelper\createDevice.jpg  Figure -SmartThings IDE Create New Device |

The Version field should remain as “Published”, and both the Location and Hub should have your location’s hub listed. Ensure both of these fields are not empty, however Group can remain empty. When you are happy with your entries, click the Create button. You will be taken back to the My Devices page, which should now include your newly created virtual switch. You can always click on the switch again and edit it should you need to, or you can edit some of the attributes (such as the name) in the SmartThings mobile application on your phone or tablet.

**Virtual Dimmer Installation:**

Setting up a virtual dimmer is slightly different than setting up a simulated switch or momentary tile. Both of those switches are native to SmartThings; a virtual dimmer is not. However, code is provided as part of the ***Alexa Helper*** suite that will allow you to create virtual dimmers as easy as the other virtual switches.

First, copy the code from the following GitHub location:

<https://github.com/MichaelStruck/SmartThings/blob/master/Other-SmartApps/AlexaHelper/VirtualDimmer.groovy>

Just as with the code for the SmartApps, be sure to click the Raw link, and then select all of the code (*CTRL+A*), and then copy it (*CTRL+C*) to your computer’s clipboard.

Then go to the SmartThings IDE at <http://ide.smartthings.com>. Log into the site and proceed to the link at the top labeled My Device Types. Click on this link.

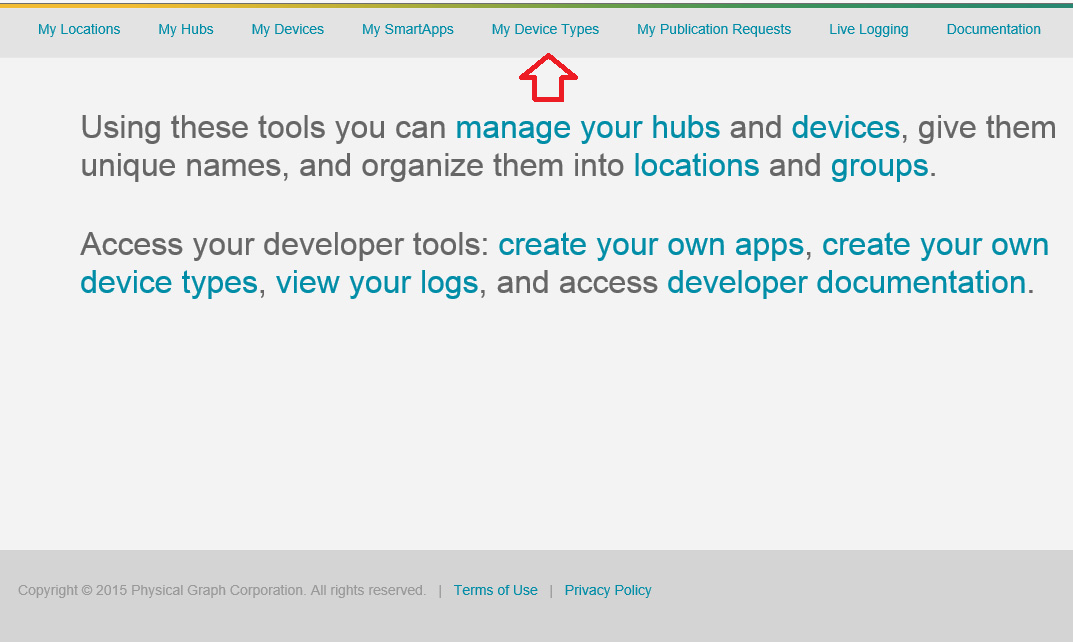
****

Figure -SmartThings IDE My Device Types

This will present you with a list of custom device types that you may have within your account. If you do not yet have the virtual dimmer listed, click the link +New Device Type in the upper right-hand corner of the screen.

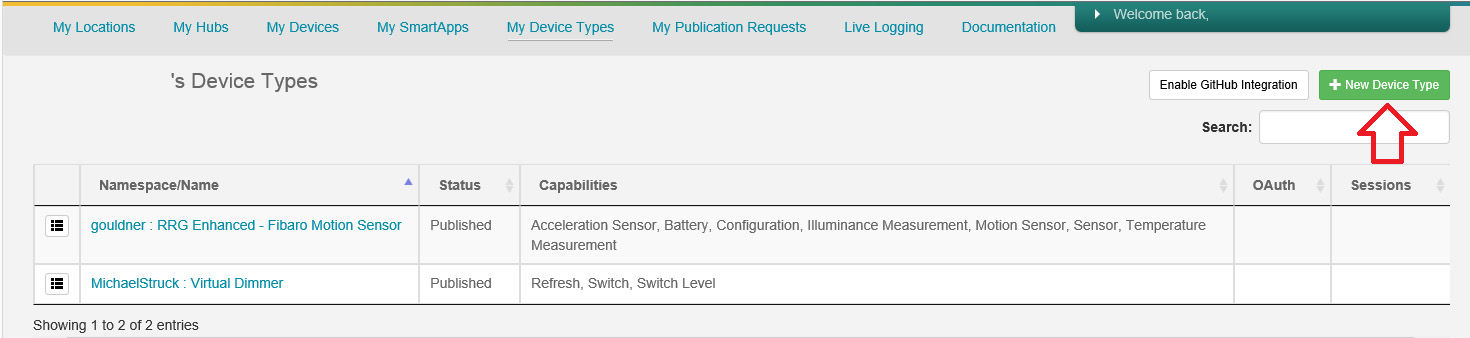
****

Figure - SmartThings IDE - My Device Type

There will be three tabs at the top, which represent the different ways to add a device. Choose From Code and you will be presented an empty area where the code you copied earlier can be placed. In the open area in the middle paste (*CTRL+V*) this code.

From here, simply Save and Publish this code, exactly as you did when creating the SmartApps.

****

Figure - SmartThings IDE – My Device Type Code

|  |
| --- |
| Notice |
| The code for the virtual dimmer only needs to be installed in this manner once; you can create multiple virtual dimmers now as long as this code is in place. |

The remaining steps for creating the virtual dimmer are the same as the other virtual switches:

* Find the section labeled My Devices at the top of <http://ide.smartthings.com>. Click this link.
* Find the button in the upper right corner of the page labeled +New Device. Click this link.

|  |  |  |  |
| --- | --- | --- | --- |
| * Fill out the fields as you did for the virtual switch, being mindful of what you use for the Name/Label if you plan to speak it to the Amazon Echo. * For the field labeled Device Network Id, ensure the number you choose is unique to your SmartThings environment. * In the field labeled Type use the pull down and scroll to bottom of the list where you will find a new device called “Virtual Dimmer”. Choose this device type. If this device does not appear in the list, go back to the code installation section above to ensure you properly published the code. * The Version field should read “Published”. * Both the Location and Hub should have your location’s hub listed. * Group can remain empty. * Click the Create button.   You will be taken back to the My Devices page, which should now include your newly created virtual dimmer switch.  From here you should have all of the components to finally install and operate Alexa Helper. | | | **C:\Users\Michael\Desktop\ClipArt-AlexaHelper\VirtualDimmerSelection.jpg**  Figure - SmartThings New Device-Virtual Dimmer |
| **Alexa Helper Installation:**  If you installed the code properly, you should find both the ***Alexa Helper*** and ***Alexa Helper-Cloud Interface*** SmartApps located in the SmartThings Mobile Application under *Market Place>>> SmartApps>>>+MyApps*.  If you just need to control lights, routines, modes, etc. simply install the ***Alexa Helper*** Smart App. If you require more advanced functionality install the ***Alexa Helper-Cloud Interface***SmartApp. Refer to the sections below for each application on the specific setups.  If the ***Alexa Helper-Scenario*** SmartApp shows up in the list, this indicates you may have inadvertently published it. To solve this, you must delete the publication, which means deleting and re-adding the code. To do this, go back to My SmartApps page on the IDE and choose the edit button () for ***Alexa Helper-Scenario***, select all of the code (*CTRL+A*) and copy it (*CTRL+C*). Then, click the App Settings link. On this page, delete the code. Then, follow the process to create a new SmartApp, ensuring when you are done that you DO NOT publish the code…only save it.  Returning to *Market Place>>>SmartApps>>>+MyApps* should now show just show the parent app and the cloud interface (if you installed the code). | C:\Users\Michael\Desktop\ClipArt-AlexaHelper\Screenshot_20151225-165616.png  Figure - SmartThings Add SmartApps | | |

**Scenario Setup:**

When you first open ***Alexa Helper*** you will be presented with the main screen which allows you to create a new scenario, set up speaker controls and thermostat controls. Let’s start with setting up a scenario first, then move on to speaker and thermostat controls. To do this, tap the link *<Create New Alexa Scenario…>*.



Figure - Create New Alexa Scenario

|  |  |
| --- | --- |
| **Name your scenario:**  Enter the name you would like to give to the scenario. This scenario name will appear in main ***Alexa Helper*** screen.  **Choose an Alexa Switch to use…:**  This is a switch that is recognized by Alexa and is typically a momentary tile or simulated switch. Dimmers may appear in the list, however it is better to use these for thermostat and speaker controls. If the switch you choose above is a momentary tile, click this option to remove the ‘off’ options  **When switch is on:**  For the options in this section, choose the routine, mode, Smart Home Monitor status you want to have enabled when the Alexa switch chosen is turned on. You also have the option of turn on certain switches, even though this functionality is present in the native ***Amazon Echo*** mobile application.  There is also an option to run an HTTP “GET” request when the Alexa Switch is turned on. This is for advanced users. If you wish to use this option with a SmartThings location, you may also install the ***Alexa Help-Cloud Interface***SmartApp.  Finally, you have the option to set a delay (in minutes) of when you want the actions to take place after you issue a voice command. For example, if you want to enable the security, but only 5 minutes after you say *<”Alexa, turn on security mode>”*, you would set this section to 5 and after that many minutes your actions under this section would actually fire.  If you are NOT using a momentary switch, there will also be a section identical to this that allows for similar functions when the Alexa Switch is turned off.  **Restrictions:**  You can set up restrictions for when this scenario can run to allow for some creativity in your voice commands. See the Tips and Tricks section at the end of this manual for some usages around restrictions.  **Remove Scenario:**  If you wish to remove the scenario you are currently editing, simply scroll to the bottom of the page and tap the red <*Remove>* button. This will remove the scenario you are currently working on; other scenarios or other SmartApps will still be installed and operational. |  |
| **Thermostat Controls:**  Natively, the Amazon Echo cannot control thermostats. Using a virtual dimmer, you can have ***Alexa Helper*** allow your Echo to control a thermostat with your voice, issuing commands like *<”Alexa, set thermostat to 68”>*. In addition, you can use momentary tiles to turn on different modes of the thermostat such as heating, cooling and auto modes.  To set up this functionality, you first must choose a dimmer switch to use as the thermostat control. This is typically a virtual dimmer. You will also need to choose a real thermostat to control. This SmartApp has been tested with numerous community and SmartThings supported thermostats, however, not all thermostats have been tested and many not function with ***Alexa Helper***.  Because there is no feedback from the Amazon Echo on what you say outside of an ‘Ok’, there are some protections you can set to ensure you don’t set the thermostat too high or too low. For example, if you say *<”Alexa, set thermostat to 68”>* and the Echo hears 58 instead, you could find the thermostat setting the incorrect temperature of the house, possibly engaging the air conditioning instead of the heater. To prevent this, input the upper and lower setting you would like ***Alexa Helper*** to send to the thermostat. Anything above or below these settings will be ignore and changed to the limits (if you say 85 and your limit is 75, the thermostat will receive 75). If you like keeping your thermostat in Auto mode, click the <*Control when thermostat in ‘Auto’ mode>* which will send the same heating and cooling set point to the thermostat. You also can ‘sync’ the settings of the dimmer with the thermostat when the SmartApp is closed. This basically turns the virtual dimmer on and sets its level to the temperature of the thermostat set point. Then you can choose to have the thermostat go into ‘off mode’ when you turn the dimmer off. When the command to turn on the dimmer (and then the thermostat) is given, you can then have the thermostat go into a specific mode with a specific set point.  If you want to change the heating, cooling or auto modes of the thermostat, you can use momentary tiles to accomplish this. Finally, you also have the option to have some pre-designated set points sent to the thermostat when changing modes. |  |

|  |
| --- |
| Notice |
| It is highly recommended that you set the upper and lower limits of your thermostat. Alexa may misunderstand your temperature commands and cause physical damage to home heating or cooling system. As the user you take full responsibility and liability for the use of this SmartApp. |

|  |
| --- |
| Notice |
| Remember not to use the word ‘degrees’ at the end of your temperature commands. The Echo believes it is changing a dimmer switch using Alexa Helper, so adding ‘degrees’ to the end of the command will only confuse the voice recognition. |

|  |
| --- |
| Notice |
| Some thermostats have modes such as *Home* and *Away* that set automatically. ***Alexa Helper*** will only control the thermostat when the mode is set to *Home*. |

|  |  |
| --- | --- |
| **Speaker Controls:**  For connected Speakers such as Sonos, you can natively use your Echo to turn the speakers on or off. However, there is no native control to allow for volume control or changing tracks. Using Alexa Helper, a virtual dimmer and a couple of momentary tiles you can gain this functionality.  To begin, choose a virtual dimmer that you will use for not only volume control but to turn the speaker on or off. Next, choose a speaker to control. Please note that the example on the right shows “vSonos” as the virtual dimmer; within the Amazon Echo mobile app this vSonos virtual dimmer is in a group by itself called “Sonos”, so issuing the command *<”Alexa, turn on Sonos”>* will actually turn on the vSonos dimmer, which in turn turns on the Master Bedroom Sonos speaker. You will also want to ensure if you use this method to turn on or off your speaker that you remove the speaker from the ***Amazon Echo*** mobile application and include the virtual dimmer you will be using.  Similarly to the temperature controls above, you can set up the upper and lower limits of the volume in case your Echo misunderstands your commands. You may also set up an initial volume whenever the speaker is turned on.  Using momentary tiles, you may also set up a next track and previous track command. In the example to the right you would simply say *<”Alexa, turn on SonosNextTrack”>* or *<”Alexa, turn on SonosPreviousTrack”>* to change tracks. |  |

**Cloud Interface Setup:**

If you SmartThings set up includes two locations, or you need to generate unique URLs for your single location, the ***Alexa Helper-Cloud Interface*** will allow you to do this. To install this SmartApp, go to your SmartThings mobile application and navigate to *Market Place>>> SmartApps>>>+MyApps*. This will bring up a list of SmartApps that you can install. Find the ***Alexa Helper-Cloud Interface*** SmartApp and tap on it. The ***Alexa Cloud*** interface will appear.

Using the SmartApp is easy. Simply tap <*Choose Switches>* on the main interface and select the switches you want to remotely control. Tap <*Done>* and below the switch selection area a new option appears to <*Show URLs>.* Tap this area and a screen will appear with the list of switches you have chosen and the unique URL for turning these on and off

|  |  |
| --- | --- |
| **C:\Users\Michael\Desktop\ClipArt-AlexaHelper\Screenshot_20151224-120724.png**  Figure -Alexa Helper-Cloud Interface | C:\Users\Michael\Desktop\ClipArt-AlexaHelper\Screenshot_20151224-120822.png  Figure - Cloud Interface-Show URLs |

Depending on your mobile device type, you can long-press on the URL needed, select all of the URL text and choose copy. You can then go to ***Alexa Helper*** and paste the URL in the appropriate area.

|  |
| --- |
| Notice |
| While you can choose every switch within your account to remotely access, it is best practice to only enable the switches you will actually be using. This prevents remote access to every switch should your access token be compromised. |

If you see an error on the URL page, you may not have enabled OAuth for ***Alexa Helper-Cloud Interface*** within the IDE. Go back to the IDE (<http://ide.smartthings.com>) and follow the instructions above to enable OAuth.

OAuth is important in the usage of ***Alexa Helper-Cloud Interface*** as it allows for access to your devices and provides adequate security to ensure access is only possible by those that have the proper access token. It may be necessary, however, to sometimes revoke the access token and re-generate it. If you ever suspect someone else has your access token, you should take the following steps.

Open ***Alexa Helper-Cloud Interface*** and find the <*Security Options>* of the main interface. Tapping this will bring up a confirmation page. Tapping <*Reset Access Token>* will generate a new Access Token. Tapping <*Done>* on this screen and on the previous screen will take you to the main interface page.

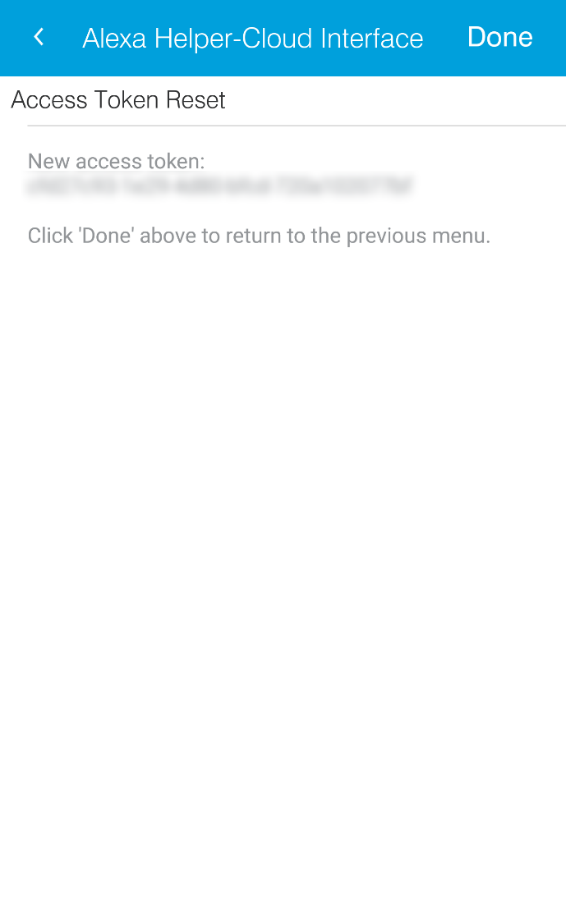
**

Figure - Cloud Interface-Access Token Reset

Please note that any time you reset your access token you will need to copy/paste the URLs into ***Alexa Helper*** or any other location that you have used these links. The old links will no longer work.

At any time you need to view your access token, or you need to determine the version number of the SmartApp you are using, tap on <*About Alexa Helper-Cloud Interface>* on the main page of the SmartApp. Again, if you see an error on this page regarding OAuth, ensure OAuth is enabled from the SmartThings IDE.

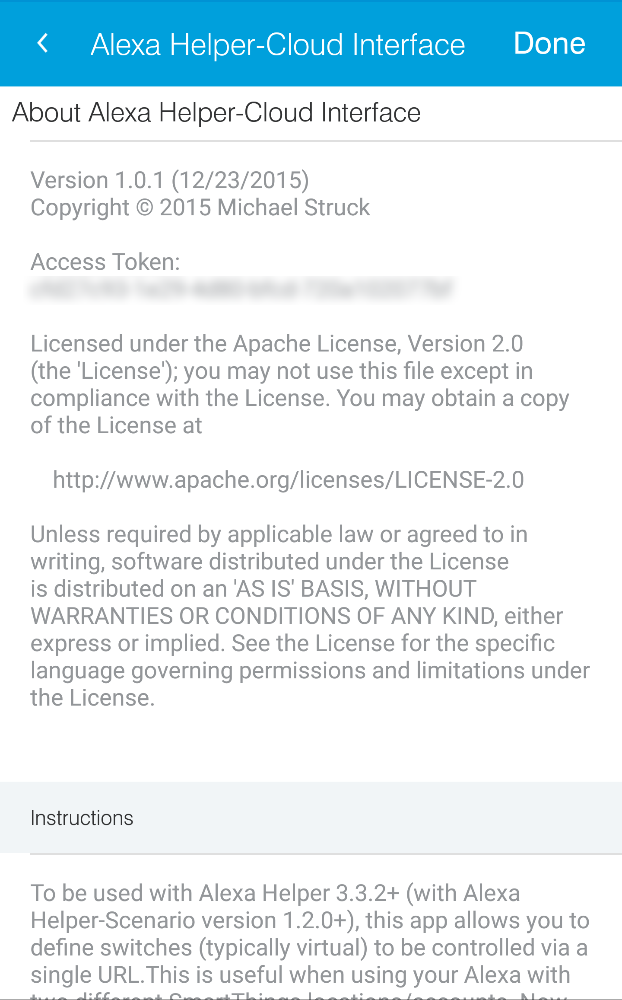


Figure - Cloud Interface-About

If you wish to remove ***Alexa Helper-Cloud Interface*** simply scroll to the bottom of the About page and tap the red <*Remove>* button. This will only remove the ***Alexa Helper-Cloud Interface*** SmartApp; it will not remove the main ***Alexa Helper*** SmartApp. However, if you do remove ***Alexa Helper-Cloud Interface*** you will no longer have remote access to the switches through the URL links.

**Tips and Tricks:**

The following are tips and tricks you can use to get the most out of ***Alexa Helper*.**

* Using the ***Alexa Helper-Cloud Interface*** SmartApp, you can generate URLs that can simply be used in a web browser. Simple copy the links you obtain from <*Show URLs>* and create bookmarks to easily initiate switch on/off action right from your computer’s browser.
* Using the Scenarios restrictions you can use a single voice command to have multiple outcomes. For example, let’s say you set a different mode on the weekdays than on the weekends when you go to bed. To accomplish this you could set up two scenarios called “Goodnight”; one that runs weekdays and one that runs weekends. This would allow the same command to be issued nightly but behave differently depending on the day of the week.
* Certain key words on the Amazon Echo are reserved and may not function if they are heard. For example, “Good Night” will not work, or “Going out of town”. In these situations, it is sometimes better to name the virtual switch something unique like “Night Mode” or “Out of town mode”.