



Stringify Developer Module (SDM)

Rev 0.3
June, 2017

OVERVIEW	3
PREREQUISITES	3
INSTALLING THE STRINGIFY DEVELOPER MODULE (SDM)	4
OPTION 1. INSTALL THE RASPBERRY PI IMAGE	4
OPTION 2. INSTALL SDM MANUALLY	5
<i>Installing Node.js</i>	5
<i>Installing the Stringify Developer Module</i>	10
<i>Optional Install – Audio support</i>	10
LAUNCHING THE DEVELOPER MODULE	11
DEVELOPER TEMPLATE	14
RPI-GPIO	14
SPEAKER	15
LET'S BUILD A FLOW.	17
USING THE GPIO PORTS ON THE RASPBERRY PI.	18
GPIO FLOW	21
DEVELOPER TEMPLATE	23
OVERVIEW	25
USING THE DEVELOPER TEMPLATE TO HANDLE EVENTS.....	26
<i>this.action</i>	26
THIS.THINGADD.....	31
THIS.FLOWADD	32
THIS.THINGDEL	32
THIS.FLOWDEL	32
USING THE DEVELOPER TEMPLATE AS A TRIGGER.	32

Overview

Stringify already connects to more than 600 amazing products and services, but we recognize that some users want to connect even more Things, and have greater control of them.

In recognition of the growing maker and DIY community, Stringify is proud to announce a [Node.js](#) module that allows more advanced users to create and connect custom Things to our platform. We are calling it the Stringify Developer Module, or SDM for short.

With this module, you can connect just about anything that runs Node.js to Stringify. You can send and receive events and use your custom Thing in flows just like any other Thing.

The Developer module connects to Stringify and lets you write custom code in JavaScript to do just about anything you can imagine.

SDM can be installed on just about anything that runs Node.js.

Prerequisites

Before you install and use the Stringify Developer Module, you should be comfortable with:

- Basic Linux command line usage
- Using command line tools such as wget, tar (although you can cut and paste almost all the commands needed from this document)
- Building Stringify Flows
- Basic hardware wiring (if working with a Raspberry Pi)
- Writing JavaScript programs (if you intend to use the Developer Template plug-in)

Installing the Stringify Developer Module (SDM)

There are two ways to install SDM:

- Option 1. If you plan on installing SDM on a Raspberry Pi, the simplest method is to download the entire SDM Raspberry Pi image and all the necessary files (Node.js, SDM, etc.) will be installed.
- Option 2. If you plan on installing SDM on a device other than a Raspberry Pi, or have a Raspberry Pi you'd rather not wipe, then option 2 is for you.

Option 1. Install the Raspberry Pi image.

Requirements:

RPI model 3
16GB SD card minimum

Download the Raspberry Pi image here:

<https://cdn.stringify.com/developer/stringify-pi.img.zip>.

Follow the instructions to flash the image to a MicroSD card based on your operating system:

Linux:

<https://www.raspberrypi.org/documentation/installation/installing-images/linux.md>

Mac OS:

<https://www.raspberrypi.org/documentation/installation/installing-images/mac.md>

Windows:

<https://www.raspberrypi.org/documentation/installation/installing-images/windows.md>

Visit <http://stringify-pi.local> from a web browser on your local LAN.

That's it! The Stringify Developer Module is installed.

Option 2. Install SDM manually.

Installing Node.js

The first thing you'll need to do (if it is not already installed) is to install Node.js. If you've already installed Node.js, skip this section and jump to the *Installing the Stringify Developer Module* section.

Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. Node is an event-driven, non-blocking I/O model that makes it lightweight and efficient. In essence Node.js runs JavaScript programs. The Stringify Developer module is written in Node.js and therefore requires Node.js to be installed in order for it to run.

If you are installing SDM on a Raspberry Pi, there are two ways to install Node.js

1. Using apt-get (the command-line tool for working with software packages), or
2. From the Node.js web site

If you are installing Node.js on a device other than a Raspberry Pi, jump to the *Installing Node.js from the Node.js web site* section below.

Using apt-get

Open up a terminal session on your Pi, then type:

```
apt-get install node.js (and press enter)
```

And simply follow the on-screen instructions.

Installing Node.js from the Node.js web site

The latest version of Node.js can be found on the nodejs.org website in the downloads section: <https://nodejs.org/en/download/>

Simply download the correct installer for your platform.

The following instructions explain how to install Node.js on a Raspberry Pi, there are a couple of more steps than installing Node.js on Windows or Mac OS X, but it's still pretty straight forward.

First, find out what the latest version of Node.js is. You can do this by going to the nodejs.org site and going to the downloads section:

<https://nodejs.org/en/download/>

It looks something like this:

Latest LTS Version: v6.10.3 (includes npm 3.10.10)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS	Windows Installer	Macintosh Installer	Source Code
Recommended For Most Users	node-v6.10.3-x64.msi	node-v6.10.3.pkg	node-v6.10.3.tar.gz
Current			
Latest Features			

Windows Installer (.msi)	32-bit	64-bit	
Windows Binary (.zip)	32-bit	64-bit	
macOS Installer (.pkg)		64-bit	
macOS Binaries (.tar.gz)		64-bit	
Linux Binaries (x86/x64)	32-bit	64-bit	
Linux Binaries (ARM)	ARMv6	ARMv7	ARMv8
Source Code	node-v6.10.3.tar.gz		

Additional Platforms

SunOS Binaries	32-bit	64-bit
Docker Image	Official Node.js Docker Image	
Linux on Power Systems	64-bit le	64-bit be
Linux on System z	64-bit	
AIX on Power Systems	64-bit	

- Signed SHASUMS for release files
- All download options
- Installing Node.js via package manager
- Previous Releases
- Nightly builds
- Node-ChakraCore Nightly builds
- Building Node.js from source on supported platforms

LINUX FOUNDATION COLLABORATIVE PROJECTS

Report Node.js issue | Report website issue | Get Help

© 2017 Node.js Foundation. All Rights Reserved. Portions of this site originally © 2017 Joyent.

Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the Trademark Guidelines of the Node.js Foundation.

Linux Foundation is a registered trademark of The Linux Foundation.

Linux is a registered trademark of Linus Torvalds.

<https://nodejs.org/dist/v6.10.3/node-v6.10.3-linux-armv6l.tar.xz>

You'll need the Linux Binaries (ARM) version 6 link. To get the file name, simply hover your mouse on the link or right click on the link and select Copy Link to copy the link to your clipboard. The screenshots here illustrate this on a Mac running Safari, but other browsers / operating systems will have a similar functionality.

The screenshot shows the Node.js LTS download page. At the top, it says "Latest LTS Version: v6.10.3 (includes npm 3.10.10)". Below that, it says "Download the Node.js source code or a pre-built installer for your platform, and start developing today." There are four main download options: "Windows Installer" (node-v6.10.3-x86.msi), "Macintosh Installer" (node-v6.10.3.pkg), and "Source Code" (node-v6.10.3.tar.gz). To the left, there's a sidebar with "LTS Recommended For Most Users" and "Current Latest Features". Under "Additional Platforms", there are links for SunOS Binaries, Docker Image, Linux on Power Systems, Linux on System z, and AIX on Power Systems. The main content area shows download links for various platforms: Windows (32-bit, 64-bit), macOS (32-bit, 64-bit), Linux (32-bit, 64-bit), and ARM (32-bit, 64-bit). A context menu is open over the "ARMv6l" link in the Linux section, with "Copy Link" highlighted.

Now that you have the link to the file you need to download, open up a terminal session on your Pi, then type:

```
wget paste_your_link_here (and press enter)
```

The command should look something like:

```
wget https://nodejs.org/dist/v6.10.3/node-v6.10.3-linux-armv6l.tar.xz
```

Press enter, and this will download the Node.js binary.

Once downloaded, you'll need to unzip the binary. Do this by typing:

```
tar -xvf paste_your_link_here (and press enter)
```

The command should look something like:

```
tar -xvf https://nodejs.org/dist/v6.10.3/node-v6.10.3-linux-armv6l.tar.xz
```

and press enter. This will unzip your files and place them in a directory called:

node-v6.10.3-linux-armv6l

Now type:

```
cd node-v6.10.3-linux-armv6l (and press enter)
```

This will place you into this newly created directory.

Finally, copy these files to /usr/local so they are globally accessible.

Type:

```
sudo cp -R * /usr/local/ (and press enter)
```

You'll be prompted to enter your password, then press enter again.

Double check to be sure you are in the node-v6.10.3-linux-armv6l directory when you do this.

That's it! Node.js is installed.

To verify that Node.js is properly installed and you have the correct version you just downloaded, type:

```
node -v (followed by enter)
```

and you should see v 6.10.3 displayed in the terminal window.

Installing the Stringify Developer Module

Now that Node.js is installed, let's install the Stringify Developer Module. This uses Node.js' package manager (npm) to install the module.

Open a terminal window (if one is not already open) and type:

```
npm install https://cdn.stringify.com/developer/stringify-developer-1.0.11.tgz -g (followed by enter)
```

The `-g` flag tells npm to install Stringify globally. There are two ways to install npm packages: locally or globally. You choose which kind of installation to use based on how you want to use the package. Be sure to include the `-g` flag here.

Note: It is possible that this install step could fail depending upon your npm settings (keep an eye on the terminal output). If it does fail, you may have to sudo to install the Stringify Developer Module. Simply preface the npm command with sudo:

```
sudo npm install https://cdn.stringify.com/developer/stringify-developer-1.0.11.tgz -g (followed by enter)
```

You'll be prompted to enter your password, then press enter again.

Depending upon your platform and configuration, the installation may take a minute and may prompt you to install some additional files, follow the instructions as prompted.

That's it! The Stringify Developer Module is installed.

Optional Install – Audio support

Note: There is one additional install you may wish to do if you want to play audio out of the Raspberry Pi's audio port, and that is to install an application called mpg123.

[mpg123](#) is a free and open-source audio player. It supports MPEG audio formats, including MP3.

It's worth taking a moment to install this application as it will let you build flows that support sound and TTS (Text To Speech).

To install this application on a Raspberry Pi, open a terminal window (if one is not already open) and type:

```
sudo apt-get install mpg123 (and press enter)
```

You'll be prompted to enter your password, then press enter again.

That completes the Stringify Developer Module installation.

Launching the Developer Module

At this point you have all the necessary prerequisites installed to start using the Stringify Developer Module.

To launch the Stringify Developer Module, open a terminal window (if one is not already open) and type:

```
stringify (and press enter)
```

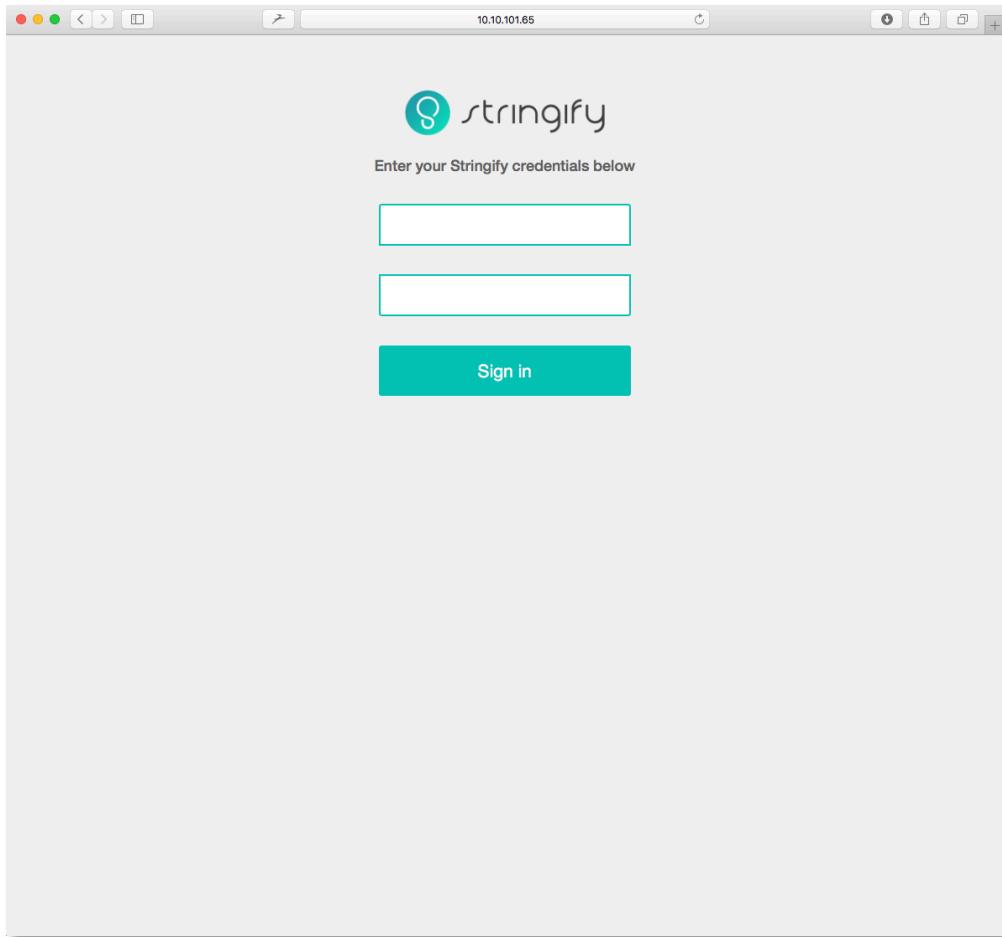
If a command not found message is displayed, navigate to /usr/local/bin and try again:

```
./stringify (and press enter)
```

Upon start-up you will see an informational message like the following, with instructions to log in via a web browser:

```
info: >>> Stringify Developer module starting up
debug: Not logged in. Please visit http://10.10.101.65:1337 in
your web browser.
```

Leave the process running, launch your web browser and go to the URL indicated; <http://10.10.101.65:1337> in this example.

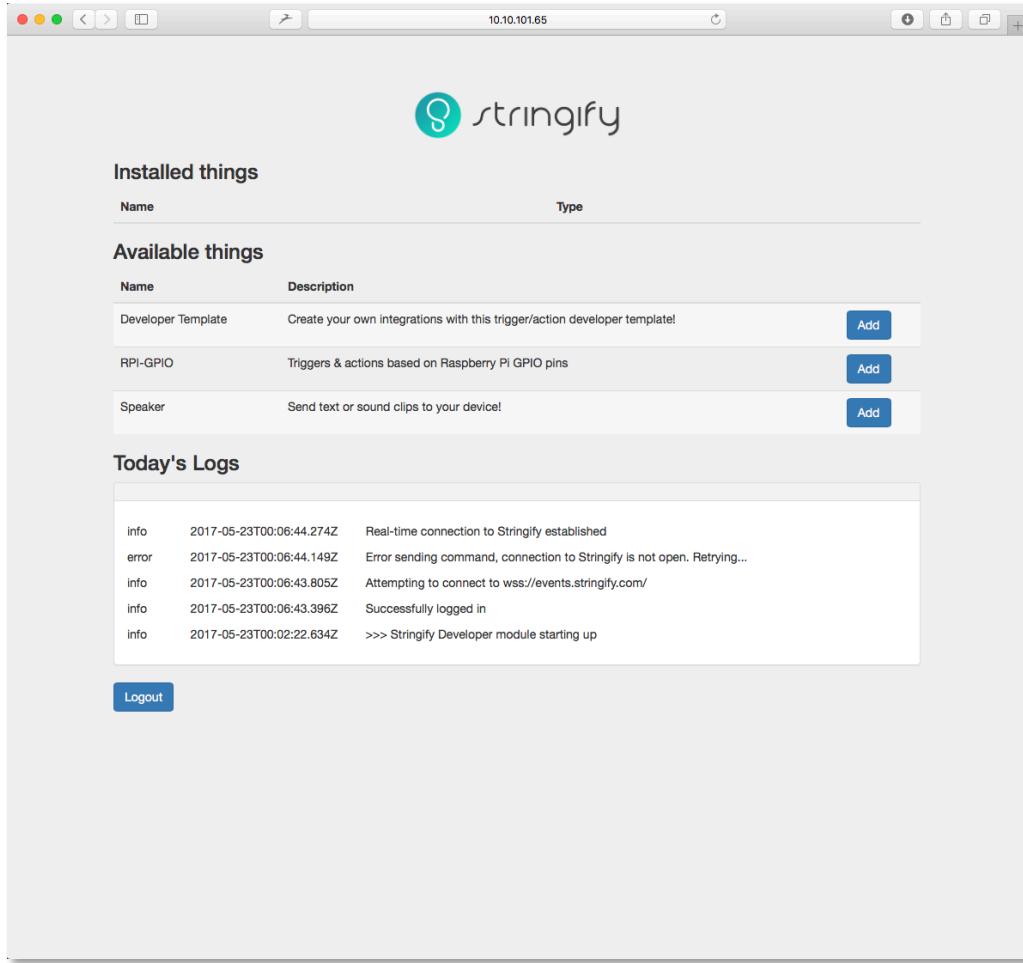


You will need a valid Stringify account to log in. If you do not have a Stringify account, use the iOS or Android client to create one.

Once logged in, the Stringify process in your terminal window will display logging information:

```
debug: Connection to browser established
info: Successfully logged in
debug: Testing user credentials...
debug: Please visit http://10.10.101.65:1337 in your web browser
to configure.
info: Attempting to connect to wss://events.stringify.com/
debug: Requesting list of your Things from Stringify for device
with mac ac:de:48:00:11:22
debug: Requesting list of available Things
error: Error sending command, connection to Stringify is not
open. Retrying...
info: Real-time connection to Stringify established
debug: Requesting list of your Things from Stringify for device
with mac ac:de:48:00:11:22
debug: Requesting list of available Things
debug: Received 0 Things in your Stringify library
debug: Received 3 available Things
debug: Received 0 Things in your Stringify library
debug: Received 3 available Things
debug: Sending browser reload request
debug: Connection to browser established
```

and the web browser will display a Stringify welcome screen with Available Things:



The Stringify Developer Module uses a plug-in architecture so features can be easily added. There are three default plug-ins.

Developer Template

Use this to create your own integrations into Stringify. It supports Triggers and Actions and lets you write custom code.

RPI-GPIO

Use this to create triggers & actions using the Raspberry Pi GPIO pins.

Speaker

Use this to send text or sound clips through the audio port of the Raspberry Pi.

Next to each option is an Add button where you can choose which plug-ins to install based upon your needs.

The screenshot shows the Stringify developer interface on a Mac OS X desktop. The window title is "10.10.101.65". The main content area has three sections: "Installed things", "Available things", and "Today's Logs".

- Installed things:** A table with columns "Name" and "Type". It currently contains one row: "Speaker" (Type: Device).
- Available things:** A table with columns "Name" and "Description". It lists three items:
 - "Developer Template": Description: "Create your own integrations with this trigger/action developer template!"
 - "RPI-GPIO": Description: "Triggers & actions based on Raspberry Pi GPIO pins"
 - "Speaker": Description: "Send text or sound clips to your device!"
- Today's Logs:** A log window showing the following entries:

```
info 2017-05-23T00:06:44.274Z Real-time connection to Stringify established
error 2017-05-23T00:06:44.149Z Error sending command, connection to Stringify is not open. Retrying...
info 2017-05-23T00:06:43.805Z Attempting to connect to wss://events.stringify.com/
info 2017-05-23T00:06:43.396Z Successfully logged in
info 2017-05-23T00:02:22.634Z >>> Stringify Developer module starting up
```

A red circle highlights the "Add" button for the "Speaker" item in the Available things section. Below the logs, there is a "Logout" button.

Once you click on Add, the Stringify process will indicate the addition of the plug-in via its logs. In the following example, we've clicked on the Speaker plug-in to Add it:

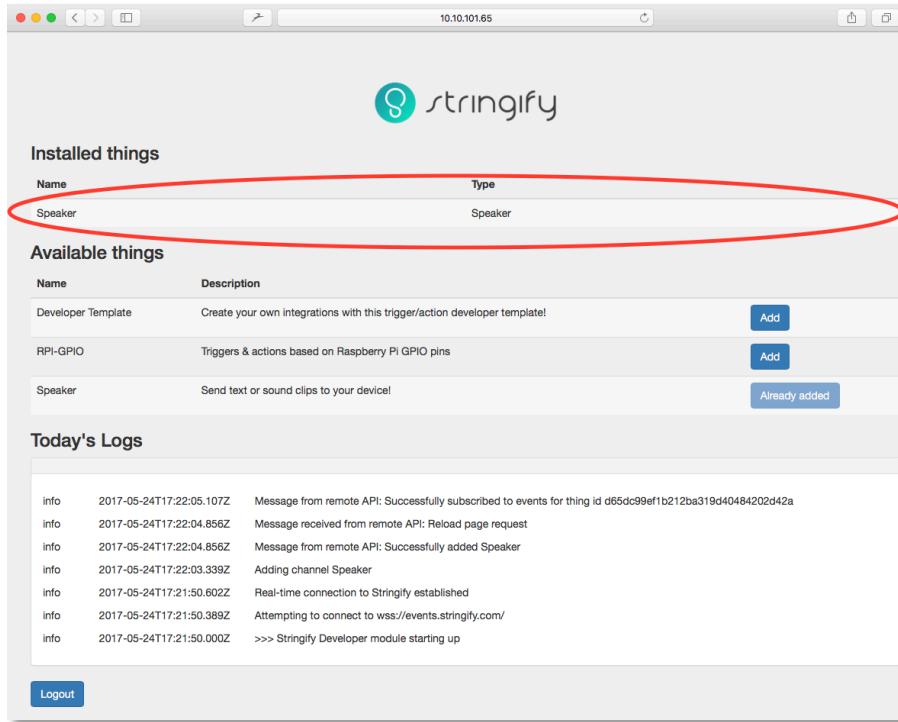
```
info: Adding channel Speaker
info: Message from remote API: Successfully added Speaker
info: Message received from remote API: Reload page request
```

```
debug: Requesting list of your Things from Stringify for device  
with mac ac:de:48:00:11:22  
debug: Requesting list of available Things  
debug: Received 1 Things in your Stringify library  
debug: Received 3 available Things  
debug: Sending browser reload request  
debug: Sending browser reload request  
debug: stringify-developer-speaker: Module Initialized.  
debug: Sending request to subscribe to events for Speaker (Thing  
id d65dc99ef1b212ba319d40484202d42a)  
debug: stringify-developer-speaker: Listening for events for  
Speaker.  
debug: Returning list of things from cache  
info: Message from remote API: Successfully subscribed to events  
for thing id d65dc99ef1b212ba319d40484202d42a  
debug: Connection to browser established
```

Note that the Thing will be automatically added to your Thing library in your Stringify app on your mobile device:



and the Stringify application web page will reflect the plug-in added:



The screenshot shows the Stringify application interface. At the top, there's a navigation bar with icons for back, forward, and refresh, followed by the URL '10.10.101.65'. Below the navigation is the Stringify logo. The main content area is divided into sections: 'Installed things' and 'Available things'. The 'Installed things' section contains a table with one row: Name 'Speaker' and Type 'Speaker'. A large red oval highlights this row. The 'Available things' section contains three entries: 'Developer Template', 'RPI-GPIO', and 'Speaker'. The 'Speaker' entry has a status bar indicating 'Already added' and a blue 'Add' button. Below these sections is a 'Today's Logs' panel containing a list of log entries. At the bottom left is a blue 'Logout' button.

Name	Type
Speaker	Speaker

Name	Description	Action
Developer Template	Create your own integrations with this trigger/action developer template!	Add
RPI-GPIO	Triggers & actions based on Raspberry Pi GPIO pins	Add
Speaker	Send text or sound clips to your device!	Already added

Today's Logs

```
info 2017-05-24T17:22:05.107Z Message from remote API: Successfully subscribed to events for thing id d65dc99ef1b212ba319d40484202d42a
info 2017-05-24T17:22:04.856Z Message received from remote API: Reload page request
info 2017-05-24T17:22:04.856Z Message from remote API: Successfully added Speaker
info 2017-05-24T17:22:03.339Z Adding channel Speaker
info 2017-05-24T17:21:50.602Z Real-time connection to Stringify established
info 2017-05-24T17:21:50.389Z Attempting to connect to wss://events.stringify.com/
info 2017-05-24T17:21:50.000Z >>> Stringify Developer module starting up
```

Logout

Note: Removing the Thing from your Stringify application via your mobile device will also uninstall the plug-in from the Stringify Developer Module.

Let's build a flow.

To demonstrate how this all works, let's build the classic “Hello, World” example using a Stringify Flow.

Build a flow using a Button Thing and the newly added Speaker flow. Call it “Hello World”. It should look like this:

And the options for the Speaker Thing should look like this:

Enable your flow. The terminal window should show a confirmation:

```
debug: Sending request to subscribe to events for Speaker (Thing  
id d65dc99ef1b212ba319d40484202d42a)  
debug: stringify-developer-speaker: Listening for events for  
Speaker.  
debug: Returning list of things from cache  
info: Message from remote API: Successfully subscribed to events  
for thing id d65dc99ef1b212ba319d40484202d42a  
debug: Connection to browser established  
info: Message from remote API: You added an instance of Speaker  
(instance id qOJ2KWCK7neyQoZ3s136) to a flow  
debug: stringify-developer-speaker: Listening for events for an  
instance of the speaker in a flow.
```

Now, find your Button Thing in your Thing library in the Stringify app on your mobile device, tap on it, and you should hear “Hello, World” spoken out of your speakers.

The terminal window will also show an action received event:

```
debug: Received an action for Speaker  
debug: Writing file  
/var/folders/9r/drgbdkf12ybchqnf1zgy5hnw0000gn/T/lw0JhV1X.mp3  
debug: Playing file  
/var/folders/9r/drgbdkf12ybchqnf1zgy5hnw0000gn/T/lw0JhV1X.mp3  
debug: Done playing, deleting file
```

That's it! You've created your first flow using the Stringify Developer Module.

Using the GPIO ports on the Raspberry Pi.

Next, we'll build a simple Flow to turn on an LED on the Raspberry Pi when a button is pressed on a phone. This example can be easily extended to trigger a flow based on external events such as weather, motion, etc., but this Flow will illustrate the basic concepts.

NOTE: You will need to run the Stringify app under SUDO to access the GPIO ports on the Raspberry Pi.

First, add the GPIO plug-in to the Stringify Developer Module. Do this the same way you added the Speaker plug-in:

The screenshot shows a web browser window titled "stringify" with the URL "10.10.101.65". The page displays three main sections: "Installed things", "Available things", and "Today's Logs".

Installed things: Shows one item: "Speaker" (Type: Speaker).

Available things: Shows three items:

- "Developer Template": Description: "Create your own integrations with this trigger/action developer template!" with an "Add" button.
- "RPI-GPIO": Description: "Triggers & actions based on Raspberry Pi GPIO pins" with an "Add" button circled in red.
- "Speaker": Description: "Send text or sound clips to your device!" with an "Already added" button.

Today's Logs: Displays a log of events from May 24, 2017:

```
info    2017-05-24T17:41:46.047Z  Message from remote API: You added an instance of Speaker (instance id oHEDEyR5XVouZlYgdxHa) to a flow
info    2017-05-24T17:30:03.678Z  Message from remote API: You added an instance of Speaker (instance id qOJ2KWCK7neyQoZ3sl36) to a flow
info    2017-05-24T17:22:05.107Z  Message from remote API: Successfully subscribed to events for thing id d65dc99ef1b212ba319d40484202d42a
info    2017-05-24T17:22:04.856Z  Message received from remote API: Reload page request
info    2017-05-24T17:22:04.856Z  Message from remote API: Successfully added Speaker
info    2017-05-24T17:22:03.339Z  Adding channel Speaker
info    2017-05-24T17:21:50.602Z  Real-time connection to Stringify established
info    2017-05-24T17:21:50.389Z  Attempting to connect to wss://events.stringify.com/
info    2017-05-24T17:21:50.000Z  >>> Stringify Developer module starting up
```

Logout button is located at the bottom left of the logs section.

Once added, you'll see the GPIO plug-in listed under Installed things on the web page:

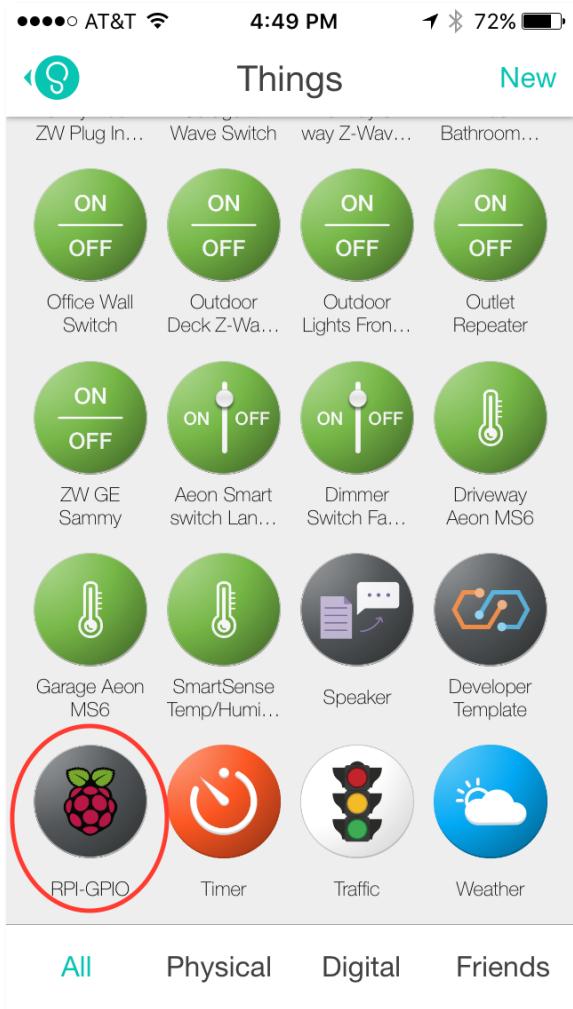
The screenshot shows the Stringify mobile app interface. At the top, there's a header with the Stringify logo and a URL field showing "10.10.101.65". Below the header, the title "Installed things" is displayed. A table lists items with columns "Name" and "Type". The "RPI-GPIO" item is circled in red and is listed under the "Type" column as "Raspberry Pi GPIO". Below this section, the title "Available things" is shown, followed by another table with columns "Name" and "Description". The "RPI-GPIO" item is listed with the description "Triggers & actions based on Raspberry Pi GPIO pins" and has a blue "Already added" button. The "Today's Logs" section is also visible at the bottom.

Name	Type
Speaker	Speaker
RPI-GPIO	Raspberry Pi GPIO

Name	Description
Developer Template	Create your own integrations with this trigger/action developer template!
RPI-GPIO	Triggers & actions based on Raspberry Pi GPIO pins
Speaker	Send text or sound clips to your device!

info 2017-05-24T17:48:31.526Z Message from remote API: Successfully subscribed to events for thing id d65dc99ef1b212ba319d40484202d42a
info 2017-05-24T17:48:31.041Z Message received from remote API: Reload page request
info 2017-05-24T17:48:31.040Z Message from remote API: Successfully added RPI-GPIO
info 2017-05-24T17:48:29.915Z Adding channel RPI-GPIO

and the GPIO Thing added to your Things library in the Stringify mobile app:



GPIO Flow

Now, we'll build a simple flow to demonstrate turning on an LED via a digital Button in the Stringify app.

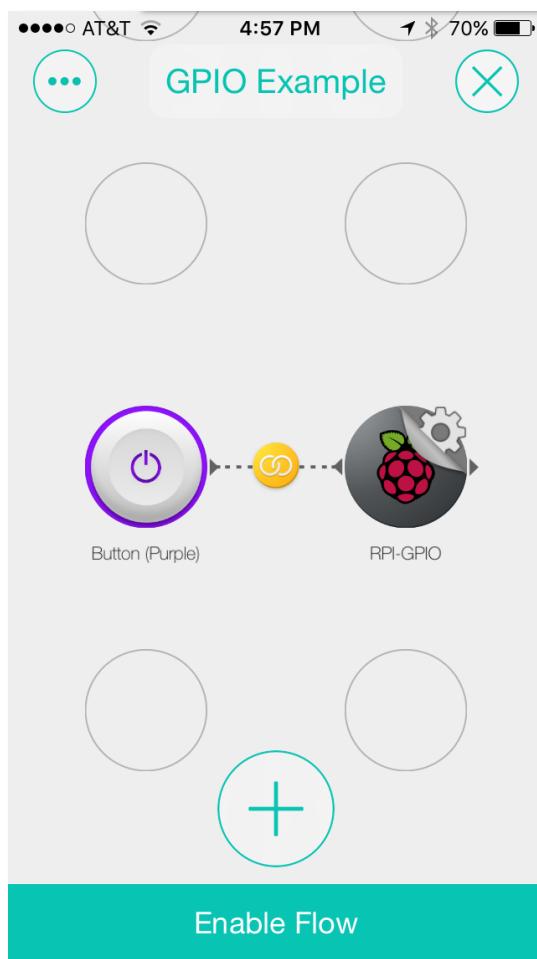
Before you can build the flow however, you'll need to make sure an LED is properly connected to the Raspberry Pi GPIO ports.

There are some excellent tutorials online for how to physically connect the necessary hardware to the Pi so this document will not go into detail on how to wire up an LED.

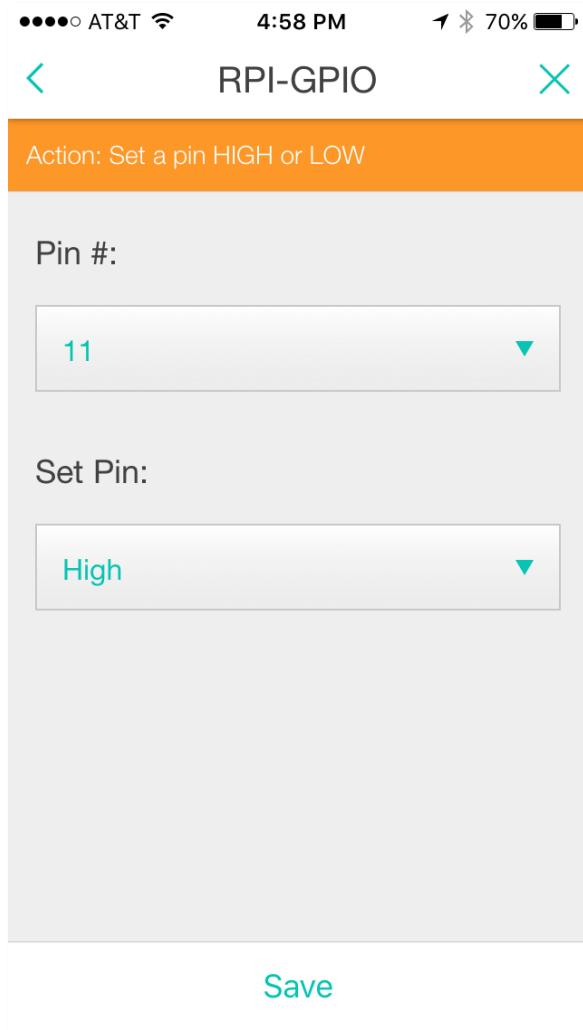
See [Gordons Projects](#) for one excellent example.

Now that your LED is wired up, let's build the flow.

Using the Stringify app on your mobile device, create a Flow using a Button Thing and the GPIO Thing as shown:



Configure the GPIO Thing settings to set pin 11 to high as shown:



Now, when you press the digital Button Thing on the Stringify mobile app, the LED on the Raspberry Pi should turn on.

If it does not, be sure to check your wiring and pin assignments.

Developer Template

The Developer Template lets you create more sophisticated interactions with Stringify using Triggers and Actions. It is arguably the more powerful of the plug-ins as it lets you write custom JavaScript code to do exactly what you want.

Before you can begin writing any code, be sure to add the Stringify Developer Template via the web interface:

The screenshot shows the Stringify web interface with the URL 10.10.101.65. It has two main sections: 'Installed things' and 'Available things'. In the 'Installed things' section, there is one item: 'Developer Template' (Stringify Developer Template). This item is circled in red. In the 'Available things' section, there are three items: 'Developer Template' (Description: 'Create your own integrations with this trigger/action developer template!'), 'RPI-GPIO' (Description: 'Triggers & actions based on Raspberry Pi GPIO pins'), and 'Speaker' (Description: 'Send text or sound clips to your device!'). The 'Developer Template' row in the 'Available things' section also has a circled 'Already added' button.

Once added, open up a terminal window and locate the directory *stringify-developer-template*. It will be located where you installed SDM. In our installation, on a Mac, it was found in:

/usr/local/lib/node_modules/stringify-developer/node_modules/stringify-developer-template

In this directory, you will find a JavaScript file called index.js:

```
drwxr-xr-x    6 nobody  staff   204 Jun  1 10:34 .
drwxr-xr-x  243 root    staff  8262 May 22 16:55 ..
-rw-r--r--    1 nobody  staff    98 Mar 14 07:37 .npmignore
-rw-r--r--    1 nobody  staff    98 Mar 14 08:31 README.md
-rw-r--r--    1 nobody  staff   2451 Jun  1 10:13 index.js
-rw-r--r--    1 nobody  staff   2419 May 22 16:54 package.json
```

This is the Developer Template file that you can edit to add your custom code to. This file handles all the communication to Stringify, so you can focus on your unique code.

In the following examples, we're using the [Sublime](#) text editor to edit index.js, but use whatever editor you are comfortable with.

Tip: If you've installed Sublime, you can create a symbolic link so you can open files directly from the command line:

```
ln -s /Applications/Sublime\  
Text.app/Contents/SharedSupport/bin/subl /usr/local/bin/subl
```

Now you can edit the Developer Template (index.js) simply by typing:

```
subl index.js
```

Overview

Now that we have the Developer Template (index.js) file open, let's take a closer look. If you are a Node.js / JavaScript developer, you should immediately be comfortable with the syntax. There are 5 functions to make note of that are involved when something "interesting" happens within Stringify. These are functions you can extend with your own code. We'll examine each briefly here, then provide more detailed examples below.

this.action

This function gets invoked when a Flow containing the Developer Template uses an Action. A string can be passed from your Flow to this function which can then be examined to determine program flow. Said another way, a Flow can pass a string to this function and then your code can determine what to do based on the value of that string.

this.thingAdd

This function gets invoked when the Developer Template is added to your library.

this.flowAdd

This function gets invoked when a Flow containing the Developer Template is enabled.

this.thingDel

This function gets invoked when the Developer Template is removed from your library.

this.flowDel

This function gets invoked when a Flow containing the Developer Template is removed.

You can add debugging code (ex. `console.log`) in the Developer Template to see what functions are called when you perform different events via web interface and mobile app.

Now let's go into each one in more detail.

Using the Developer Template to handle events.

this.action

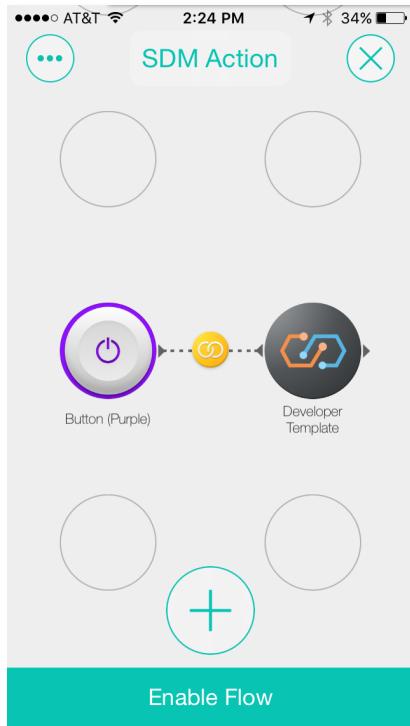
This function gets invoked when a Flow containing the Developer Template uses an Action. A string can be passed from your Flow to this function which can then be examined to determine program flow. Said another way, a Flow can pass a string to this function and then your code can determine what to do based on the value of that string.

Let's build a flow to demonstrate this function.

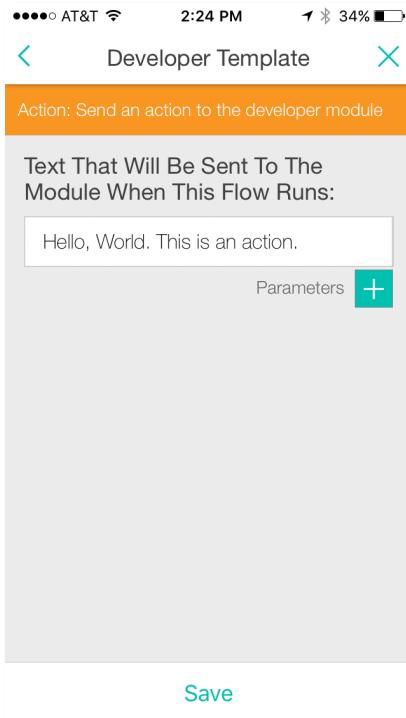
First, modify the *this.action* function to contain the following:

```
this.action = (message) => {
    console.log("Received message: " + JSON.stringify(message));
};
```

Next, using the mobile App, build a flow that uses a Button Thing to pass a string to the Developer Template. It will look something like the following:



Here we have a button (the Trigger) that when clicked will cause the Developer Template Thing to pass a string to the Node.js Developer Template process. The settings for the Developer Template Thing look like the following:



Here we have specified the string: “*Hello, World. This is an action.*” which will be passed to the Developer Template *this.action* function.

Enable the Flow, go to your Thing Library and tap on the button you added to the Flow. (Make sure Stringify is running on your device). You may need to invoke it with:

```
sudo stringify
```

You should immediately see on the console, something similar to the following:

```
debug: Received an action for Developer Template
Received message:
{"id":"bVYku5wvZwvTJx3cls4M", "flowId": "AOD1AQOvSYxlyBVI5mB2", "controls": {"action": "Hello, World. This is an action."}, "action": "action"}
```

Due to the code change we made to *this.action* above:

```
this.action = (message) => {
  console.log("Received message: " + JSON.stringify(message));
```

```
};
```

Note that the message received:

```
{"id": "bVYku5wvZwvTJx3cls4M", "flowId": "AOD1AQOvSYxlyBVI5mB2", "controls": {"action": "Hello, World. This is an action."}, "action": "action"}
```

is a JSON object that can be easily parsed:

```
▼ object {4}
  id    : bVYku5wvZwvTJx3cls4M
  flowId : AOD1AQOvSYxlyBVI5mB2
  ▼ controls {1}
    action : Hello, World. This is an action.
    action : action
```

Explanation of the fields:

id is the id of the Thing in the Flow. This can be used to uniquely identify Things in Flows.

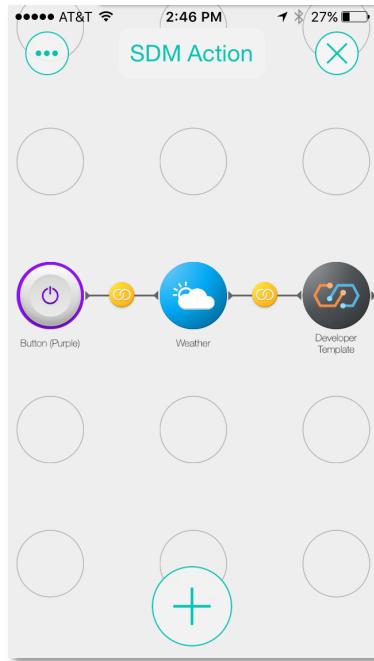
flowId is the id of the Flow. This can be used to uniquely identify Flows.

controls.action is the string you passed to the Developer Template from your Flow. This can be examined to determine what action your code should take next.

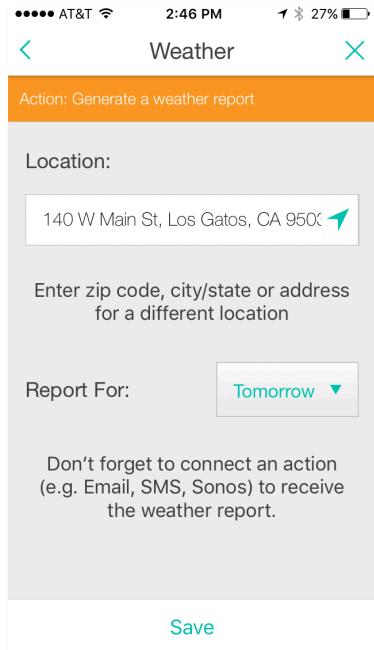
action is the type of event received. Currently there is only one.

Notice how the string you send to the Developer Template does not need be only static text as shown in our example. You can send text passed from other Things in a Flow. For example, you could send the weather temperature to the Stringify Developer Template and perform a task specific to the temperature.

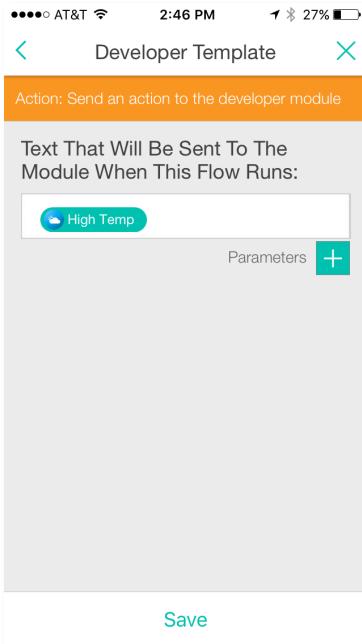
The following Flow shows an example of passing the Weather temperature to the Developer Template:



The following shows the Weather Thing Action settings:



And here we show passing the High Temp Parameter from Weather to the Developer Template:



When this Flow is run (by pressing the button that starts the Flow), it should result in the following console output:

```
debug: Received an action for Developer Template
Received message:
{"id":"lOiwSucFv3m0NzavzY3r","flowId":"AOD1AQOvSYxlyBVI5mB2","controls":{"action":"Tomorrow's high: 81F"},"action": "action"}
```

The *action* value (“Tomorrow's high: 81F”) could be examined, and program flow logic could execute based upon that value.

Note that the *action* value in this example is a full string: “Tomorrow's high: 81F” and not simply the numeric value 81. This is because the Weather Thing passes a “friendly” string so that when the Weather Thing is used with other Things such as a connected speaker, it allows the text to be read aloud properly. If you want to extract only the numeric value 81, you could place our Text Thing in front of the Developer Template in your Flow. It has a function that can be used to strip out non-numeric values from a string.

this.thingAdd

This function gets invoked when the Developer Template is added to your library.

To see this in action, add a SDM plug-in via the web interface.

[this.flowAdd](#)

This function gets invoked when a Flow containing the Developer Template is enabled.

To see this in action, add and enable a Flow via the Mobile app.

[this.thingDel](#)

This function gets invoked when the Developer Template is removed from your library.

To see this in action, remove a SDM plug-in via the Mobile app.

[this.flowDel](#)

This function gets invoked when a Flow containing the Developer Template is removed.

To see this in action, remove a flow via the Mobile app.

Using the Developer Template as a Trigger.

As you have seen, the Developer Template can respond to Actions created via Flow. It can also send Triggers that Flows respond to.

A Trigger emitted from the Developer Template contains a string that must match a string specified in a Flow.

Let's see this in action.

First, modify *index.js* to emit a Trigger. Right after the *this.flowDel* function add the following code so *index.js* looks something like:

```
•  
•  
•
```

```
this.flowDel = ...  
  
setInterval(function() {  
  console.log('Emitting trigger hi');  
  self.emit('trigger', {  
    trigger: 'I am a trigger'  
  });  
}, 5000);  
  
. . .
```

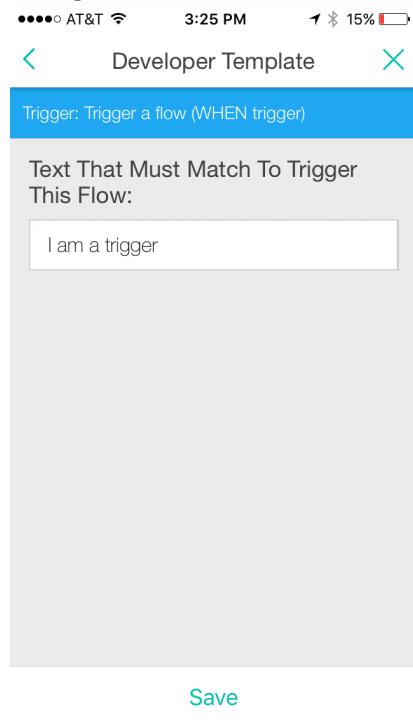
This example will cause the Developer Template to emit a Trigger every 5 seconds (5000 milliseconds) that the Stringify Flow engine will respond to.

In addition to this code change, create a Flow using the Mobile App that looks something like the following:

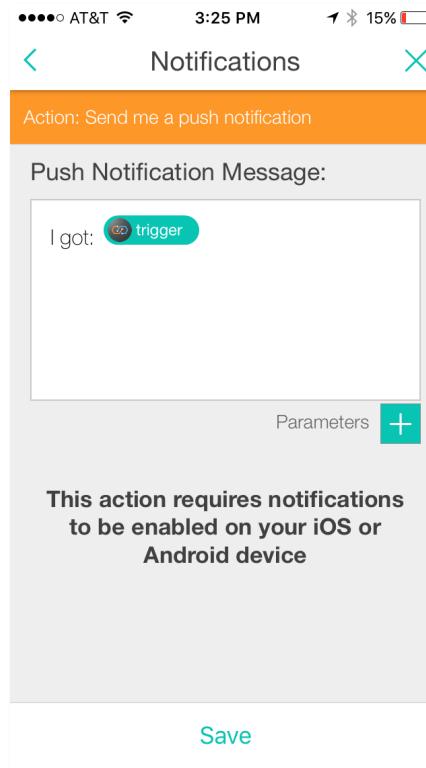


This tells the Developer Template Thing to send an Action containing a string to the Notification Thing. Of course, this could be any other Thing such as a Speaker Thing for performing Text To Speech (TTS).

The Developer Template Thing Action looks like the following:



And, finally, the Notification Thing looks like the following:



Now go ahead and run Stringify via your terminal session, and every 5 seconds you should get a notification on your phone that says “I am a trigger”. (Be sure your Flow is enabled via the Mobile App):



Be sure to disable your Flow on your Mobile App to stop receiving the notifications after you have tested this functionality.

That's it! You can now write code that sends events to Stringify that can be used in your Flows, and you can create Flows that send events to your code.

This document just touches on the very basics of what you can do with SDM. We hope this helps get you started. If you have any questions or suggestions, don't hesitate to reach out to us.