Twiddler – A Fiddler Extension for Twilio

# Overview

“Twiddler” ([actually named by Devin Rader, a Twilio Evangelist](http://www.linkedin.com/in/devinrader)) was created as a Fiddler 4.0 extension to simplify the development of Twilio applications. It allows you to receive a Twilio web hook anywhere there is an internet connection. This includes public hotspots, or even those created by your mobile phone. If you have access to the internet, your laptop via Twiddler becomes accessible from Twilio. I think it’s a fascinating leveraging of technology that still amazes me.

In brief, Twilio applications allows you to use the telephone as easily (or even easier) than you browser. In fact, with a Twilio, the scope of users that can use your application increases to anyone that knows how to use a phone. The possibilities are endless, and we are only at the tip of the iceberg!

Twilio works via a [webhook](http://en.wikipedia.org/wiki/Webhook), which turns a phone ringing into a web POST/GET to a web server of your choice. With this web POST/GET, enough information is received so that a web application can react and build an IVR system (Interactive Voice Recognition).

The catch is that a webhook requires a “publicly accessible web server”. That means a web server that is typically hosted in the cloud. However in development, we typically don’t have a public webs server. Typically we are working with a web server on our workstation, and it’s behind NAT routers and firewalls. The job of “routing in a request” from the internet is not trivial as it often requires configuration of routers. This involves Port forwarding, or protocols (like TURN/STUN servers) to manage the routing. This all requires a learning curve that typically makes a simple investigation into an overly long process that is simply not that fun, and is typically out of the realm of a hobbyist developer.

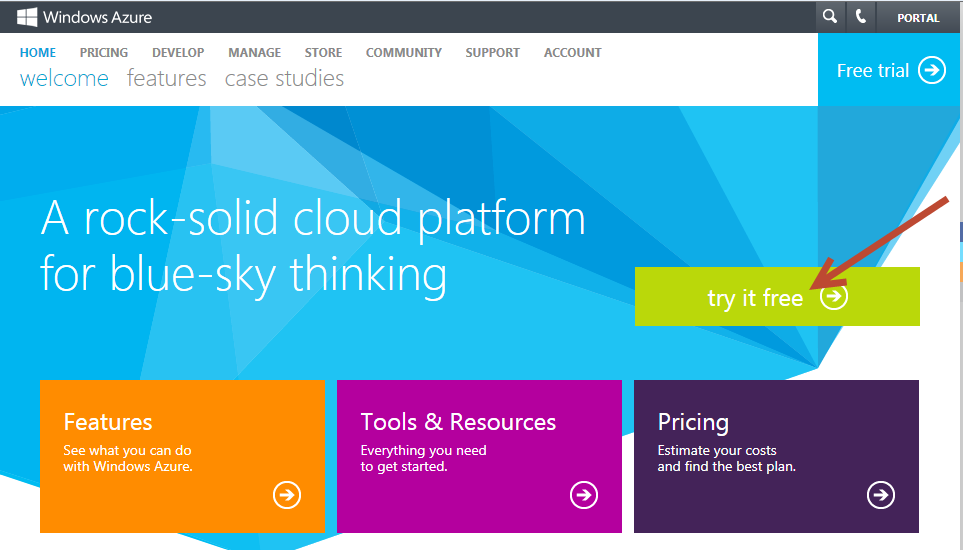
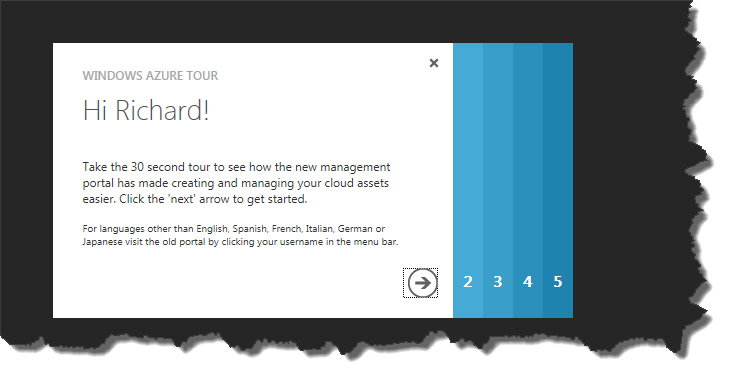
I build Twiddler because I am a consultant, and I’m always traveling with my laptop. If I wanted to do a bit of develop using Twilio, I was unable to do so because I was away from my home resources where I have configured my hardware appropriately. I wanted something so that I could turn on my laptop, and start working regardless of where I am, and start receiving those webhooks.

Twiddler does that for me via the Azure Service Bus, and much, much more! This takes no configuration other than simply creating:

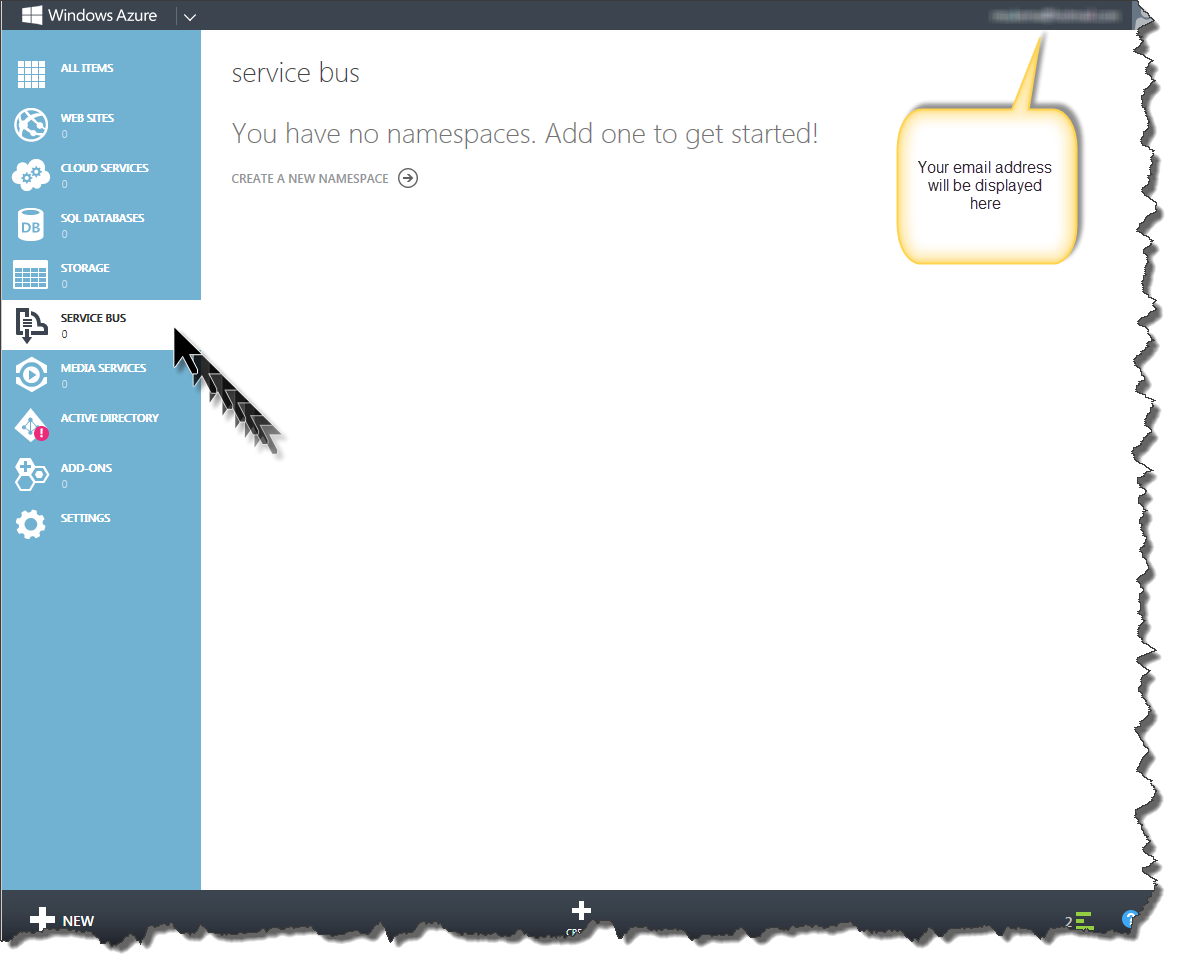
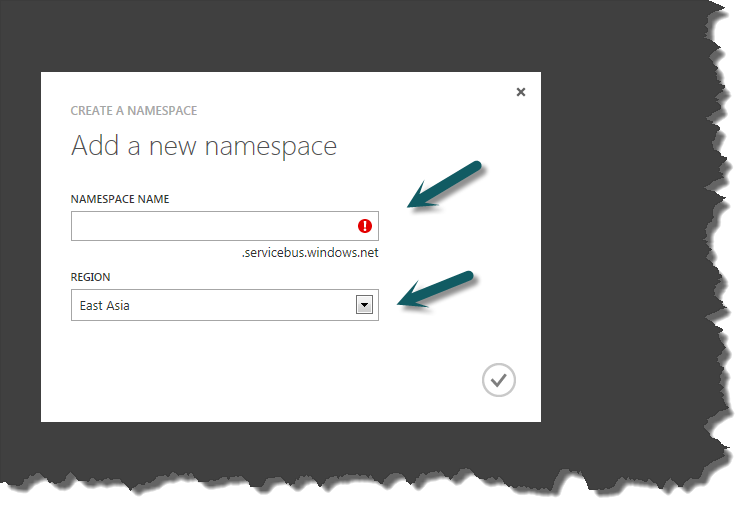
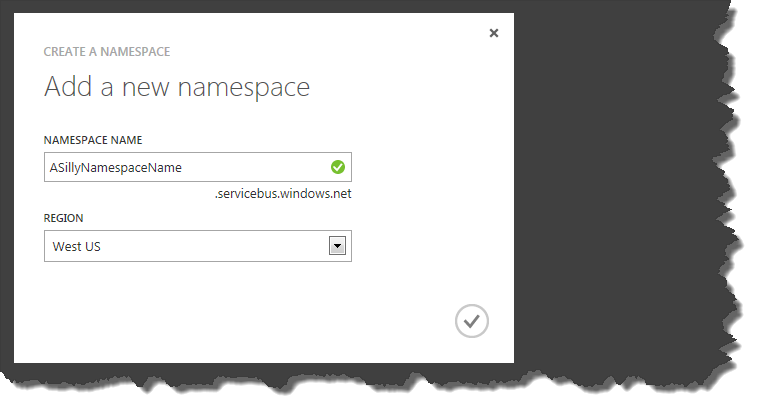
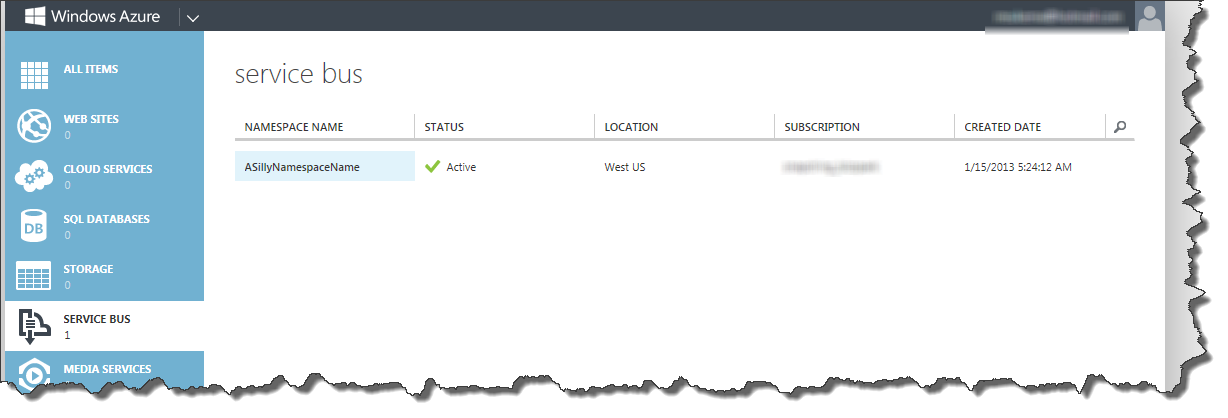
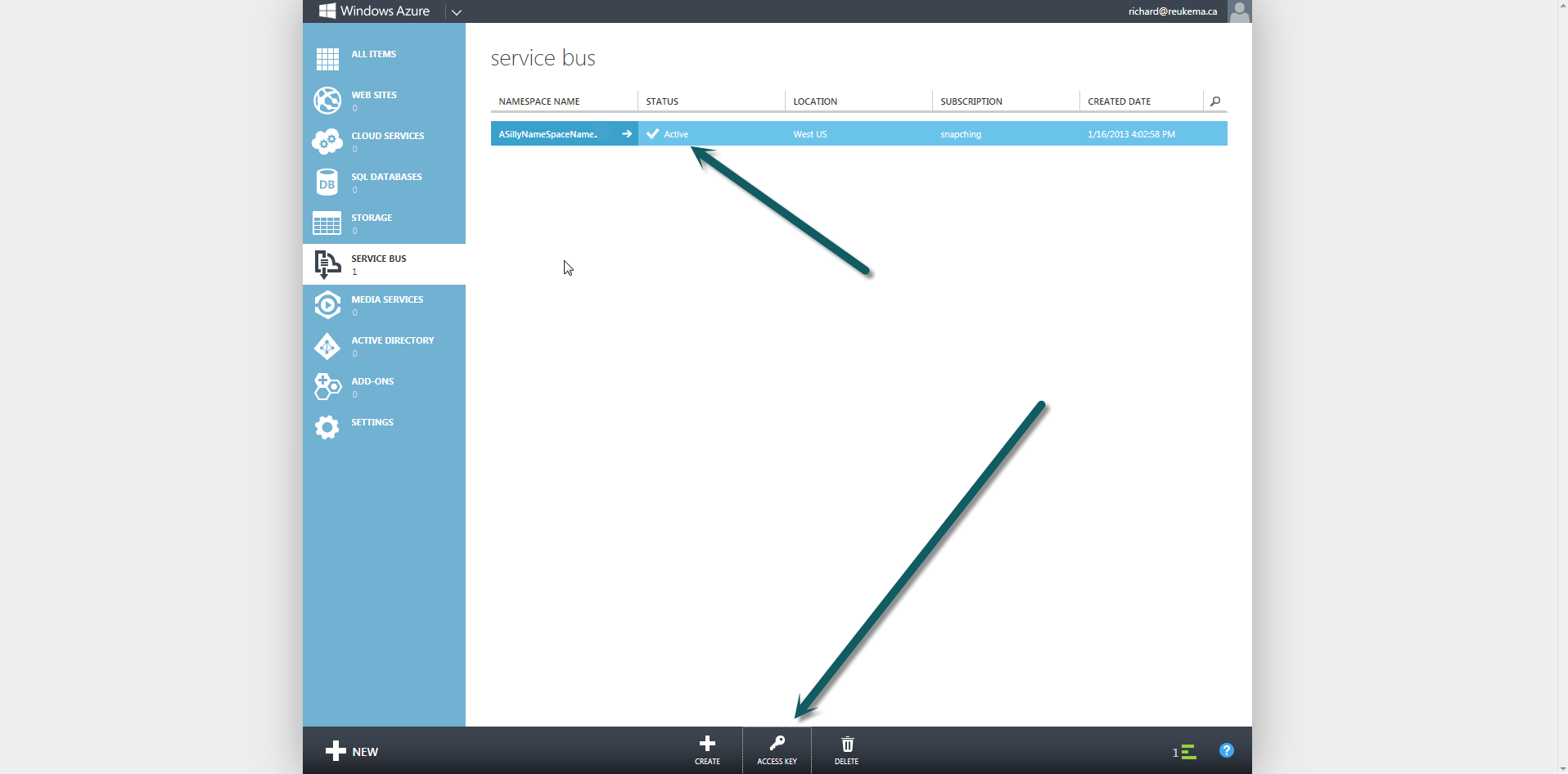
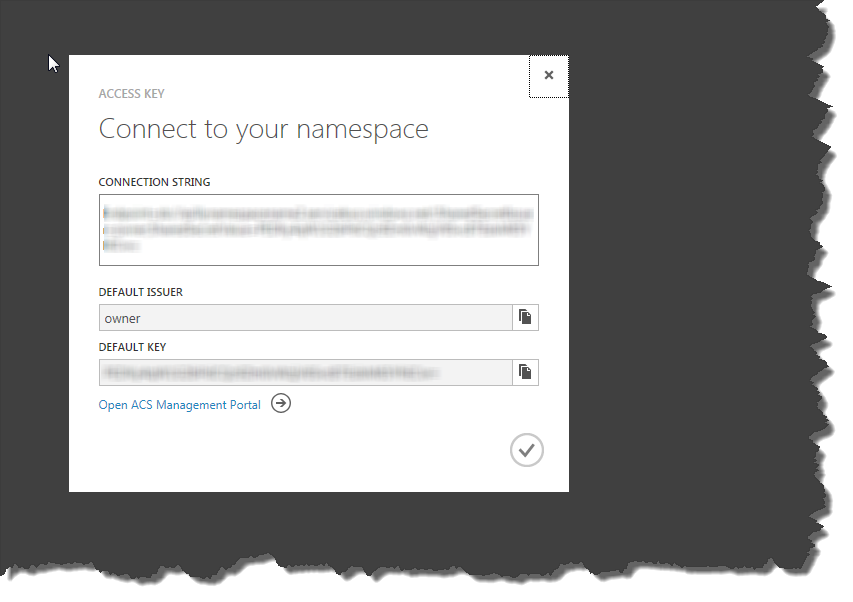
1. An Azure account (free here [WindowsAzure.com](http://www.windowsazure.com/en-us/)), and
   1. Establishing a Service Bus endpoint (detailed setup follows).
2. A Twilio Account (free [Twilio.com](http://www.twilio.com)),
3. The fabulous Fiddler 4 (free [Fiddler.com](http://www.fiddler2.com/fiddler2/version.asp)) application, and
4. Twiddler (free <https://github.com/devinrader/Twiddler>)

That may seem to be a long list for a new comer, but it’s taken me 4-6 months of a learning curve to put Twiddler together to a point that I feel is simple enough to explain in blog post.

# Azure Setup

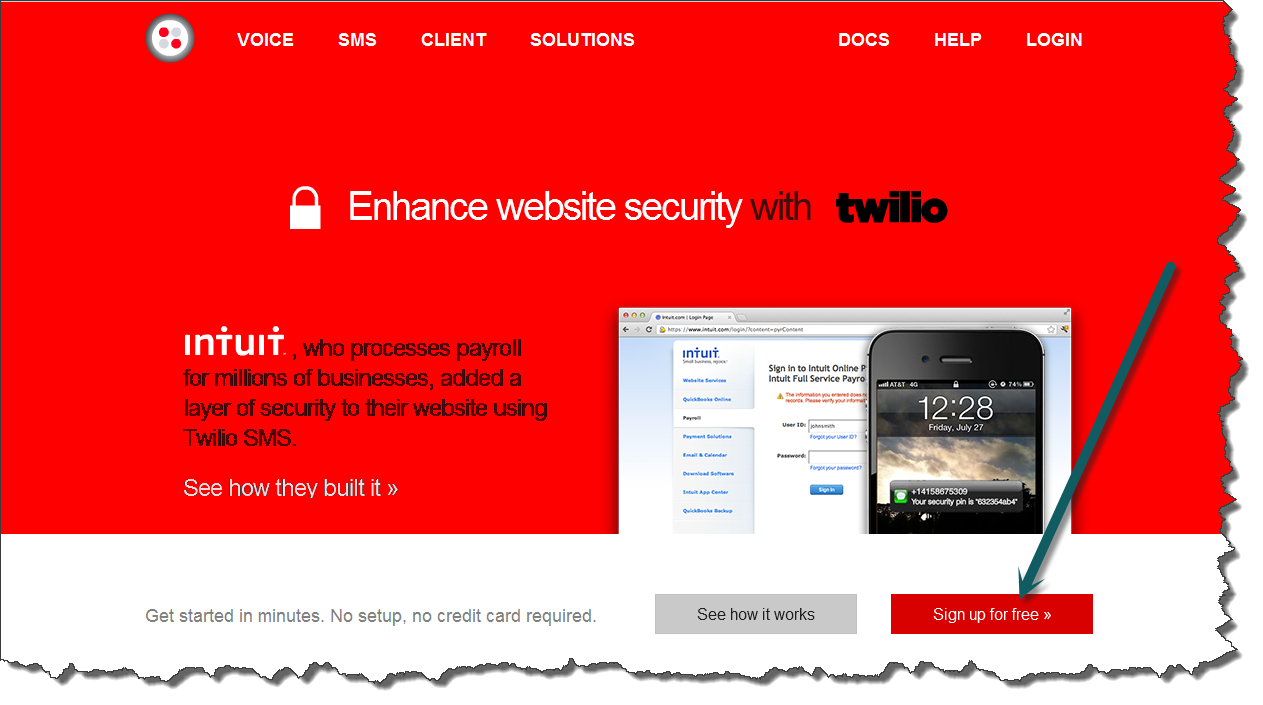
1. Create a Windows Azure Account
   1. Using a Live ID, create an Azure Account via [www.windowsazure.com](http://www.windowsazure.com)  
      
   2. After you have created an account, the intro starts  
      

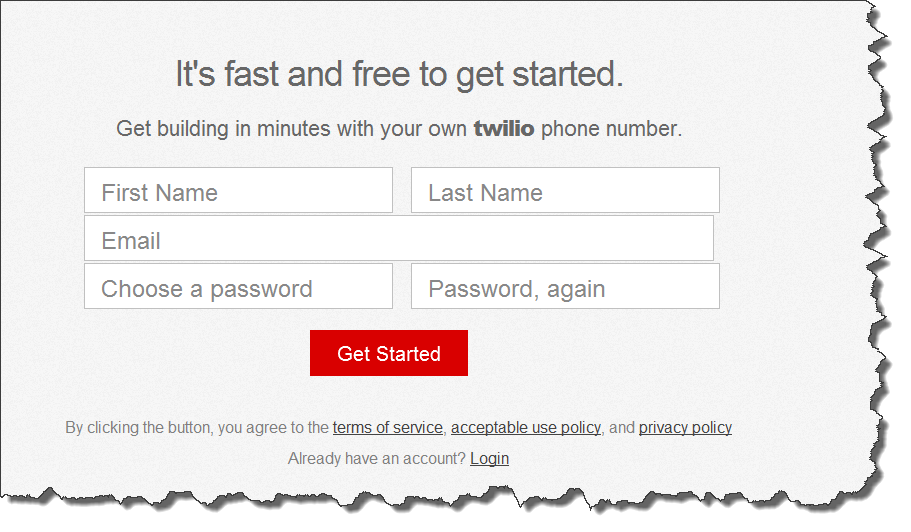
click through each of the tabs for a quick overview of Azure capabilities are.

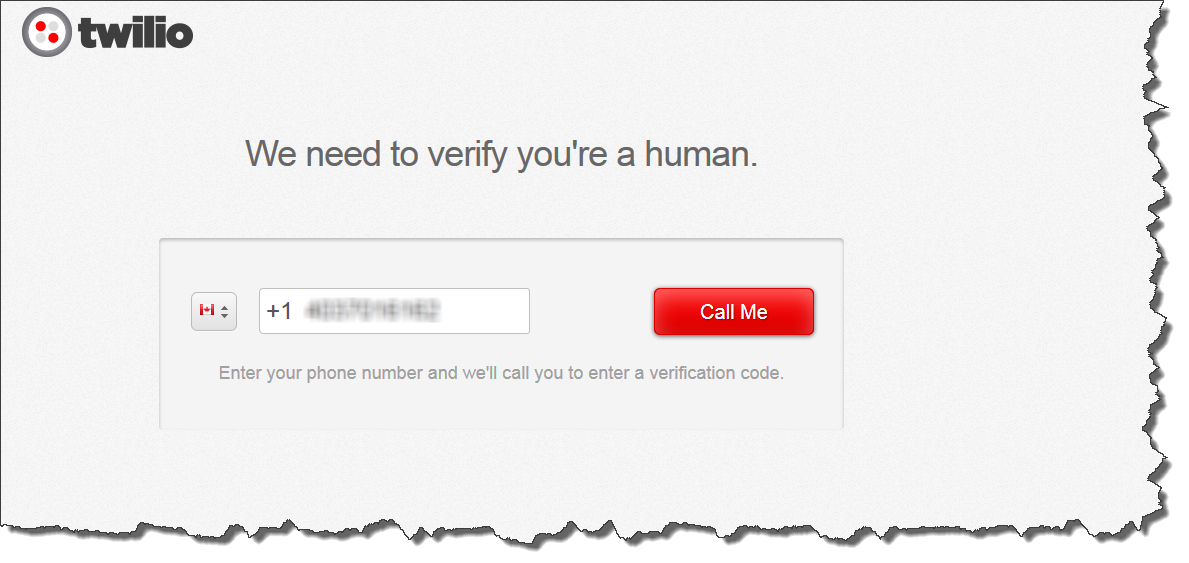
* 1. In the end you should see the main portal page  
       
     Click on the Service Bus, and click on “Create a new Namespace”
  2. New Service Bus Namespace  
     
     1. NameSpace Name  
        Enter a new namespace name. This can be anything you want, but it must be unique. The portal will validate the name, and will indicate if the name you have entered is OK (check mark will appear, see next screen shot)  
        NOTE: Directly under the NameSpace Name, the full URL will be displayed. All ServiceBus namespaces are suffixed with “.servicebus.windows.net” to create a fully specified HOST. This will be your “public” endpoint that Twilio will issue the webhook to. Twiddler (not Twilio but Twiddler the Fiddler extension) will take all activity directed to this endpoint, and “relay” it to a local workstation URL.
     2. Region  
        Select the geographical region that you are closest to.
  3. Azure Service Bus endpoint created!  
       
     and back to the portal  
     
  4. Get a Copy of the Access Key  
     The Access Key is needed later when Twiddler is configured. The key allows Twiddler to communicate with the ServiceBus and setup relay for you.
     1. Select the Service Bus NameSpace  
        This will highlight the row and activate the Access Key button. NOTE: The NameSpace must be specified as “Active” for this to work. If you just created the NameSpace, this may take 10-30 seconds.
     2. Click on the Access key near the bottom of the screen  
        
     3. Copy the “Default Key” and paste it where it can be recalled later when Twidder is being configured  
          
        Note: I’ve had to blur out the key, otherwise I may not like my credit card next month!

Your done setting up an Azure ServiceBus!   
  
NOTE: Make sure you remember the Servicebus **Namespace name**, as well as the **Default key**, as it will be needed later when we configure the Twiddler extension.

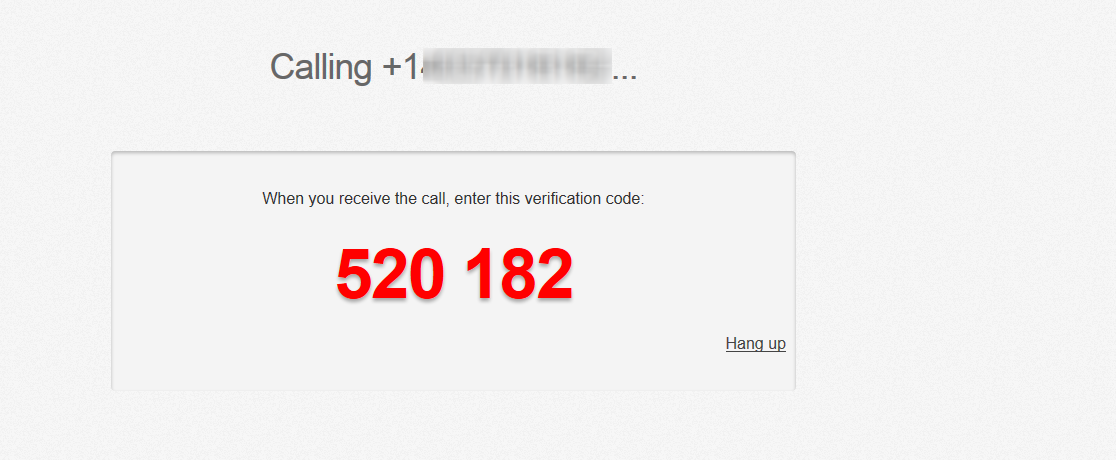
# Create A Twilio Account

1. Navigate over to Twilio.com  
   

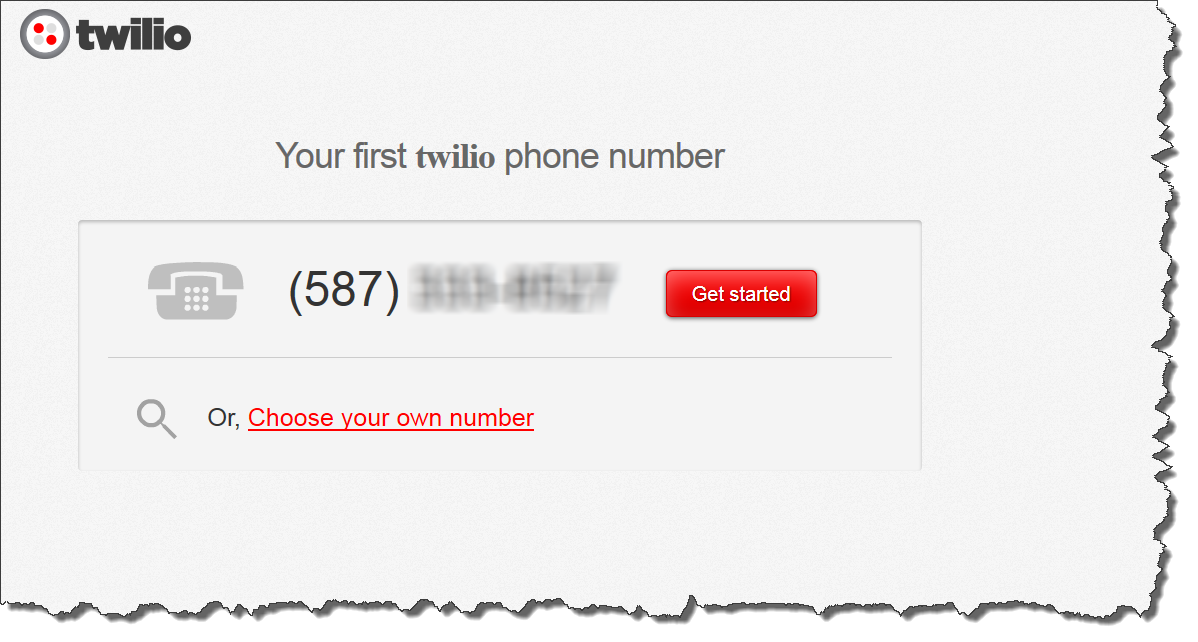
Enter your info:  


1. To verify the Account, Twilio needs to call you. Simply enter a phone number where you can be reached:  
   

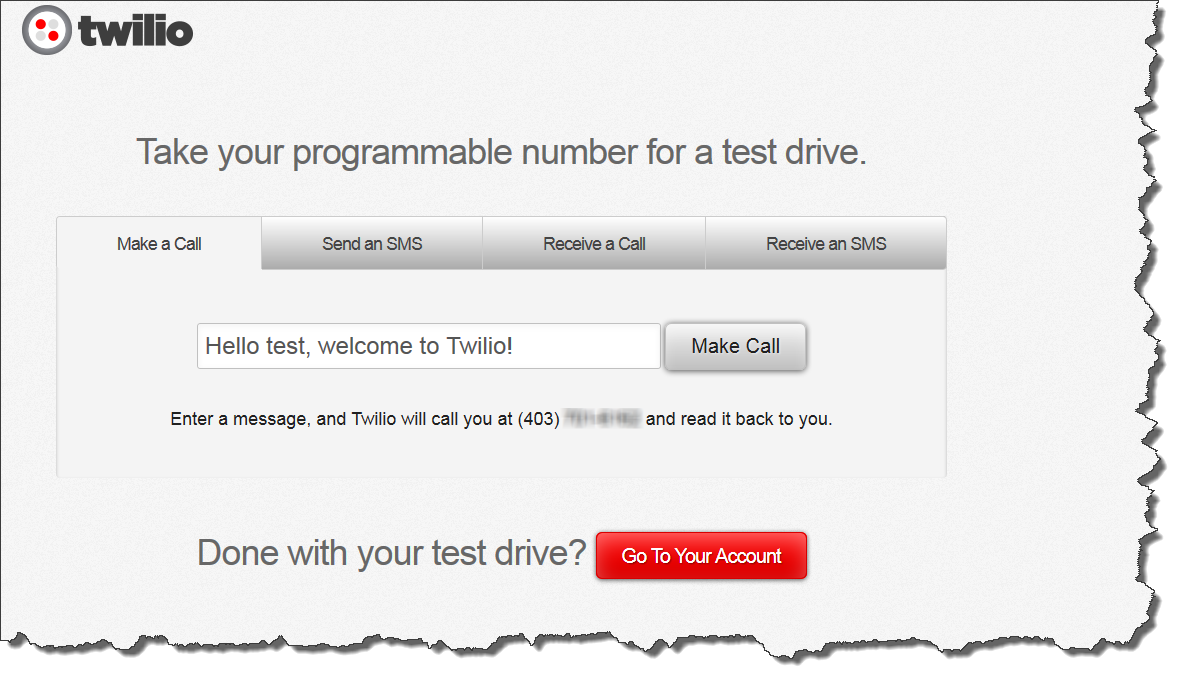
Click the “Call Me” button

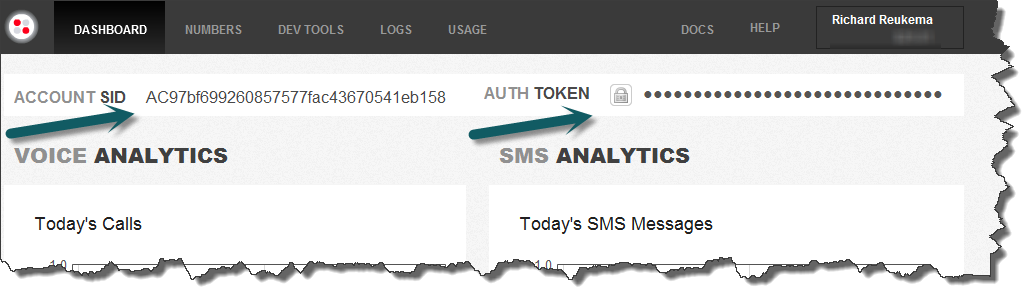
1. A verification number will display  
   

And shortly thereafter (like within 30 seconds) the phone number that you entered will ring! Listen to the instructions, and enter the verification code when prompted

and enter the verification code that was given via the web page:   
  
At this point in time, the Twilio number that you were given is active, and your almost ready to roll.

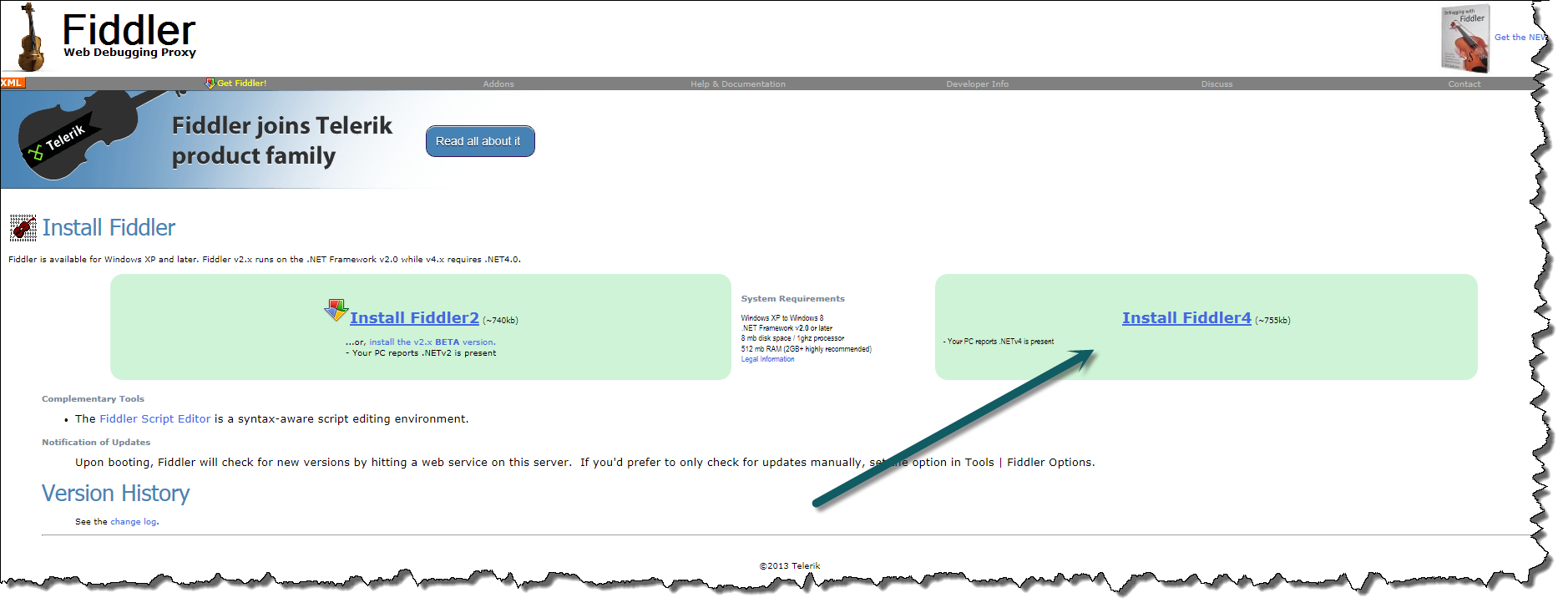
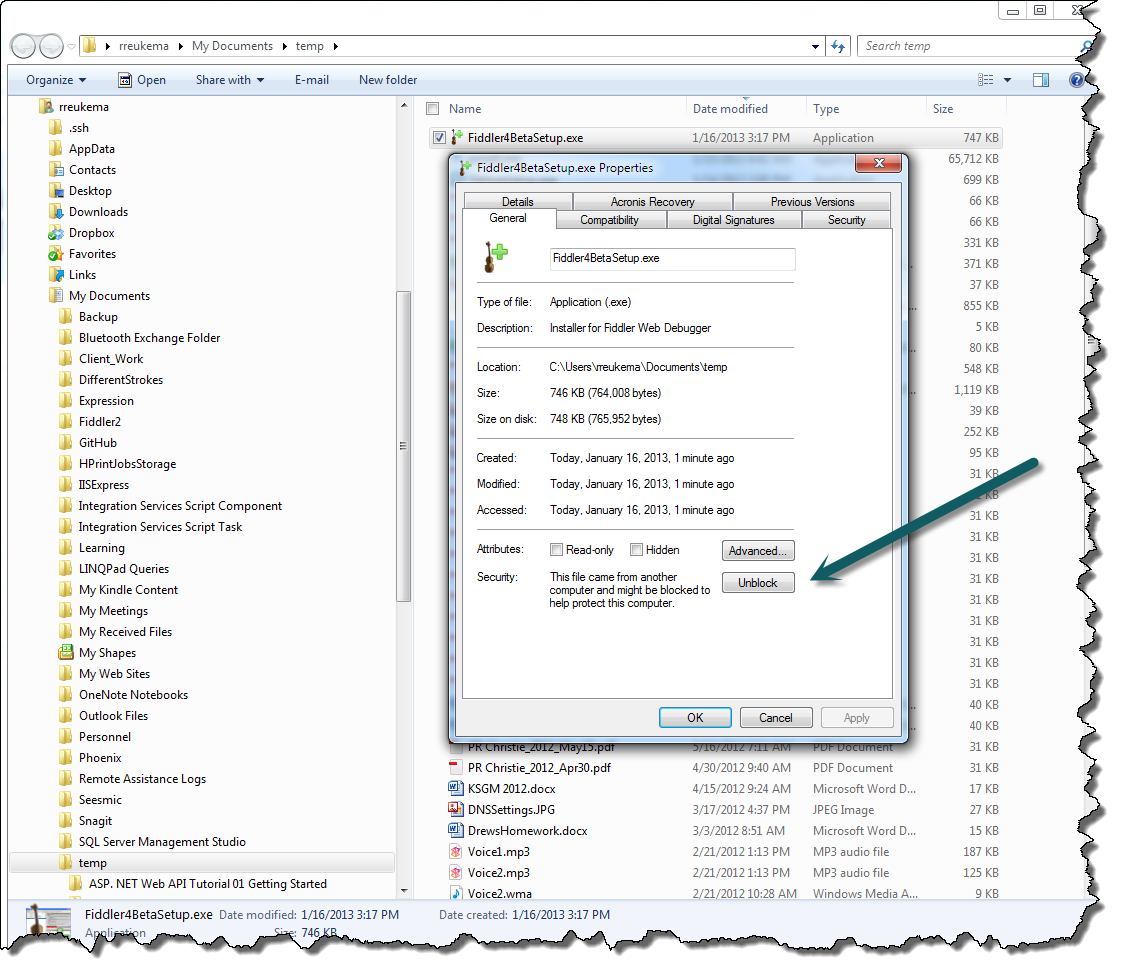
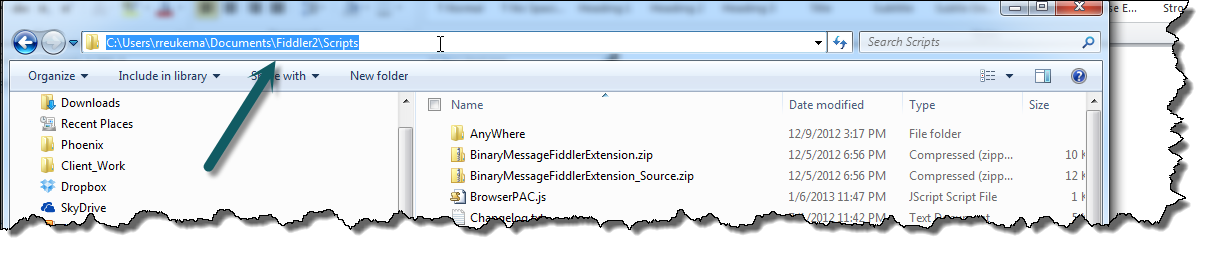
If you want to try it out, click on the “Get Started” button, and you will see the following dashboard that will allow you to “test drive” your Twilio number:

After you are done, click on the “Go To Your Account” to see the Twilio “Dashboard” for your account.

1. On the Twilio DashBoard, each Twilio account is given an:
   1. Account SID  
      A unique identifier for the an account with Twilio
   2. Authorization Token  
      A unique “password” that, along with the Account SID, allow you to use the Twilio API.  
        
      NOTE: For security reasons, the Auth Token should never be displayed publically. Click on the Lock icon to see the actual number. You will need return to this page (DashBoard via the menu bar at top) as you will need to cut and past the Account SID, and Authorization Token into Twiddler later on.

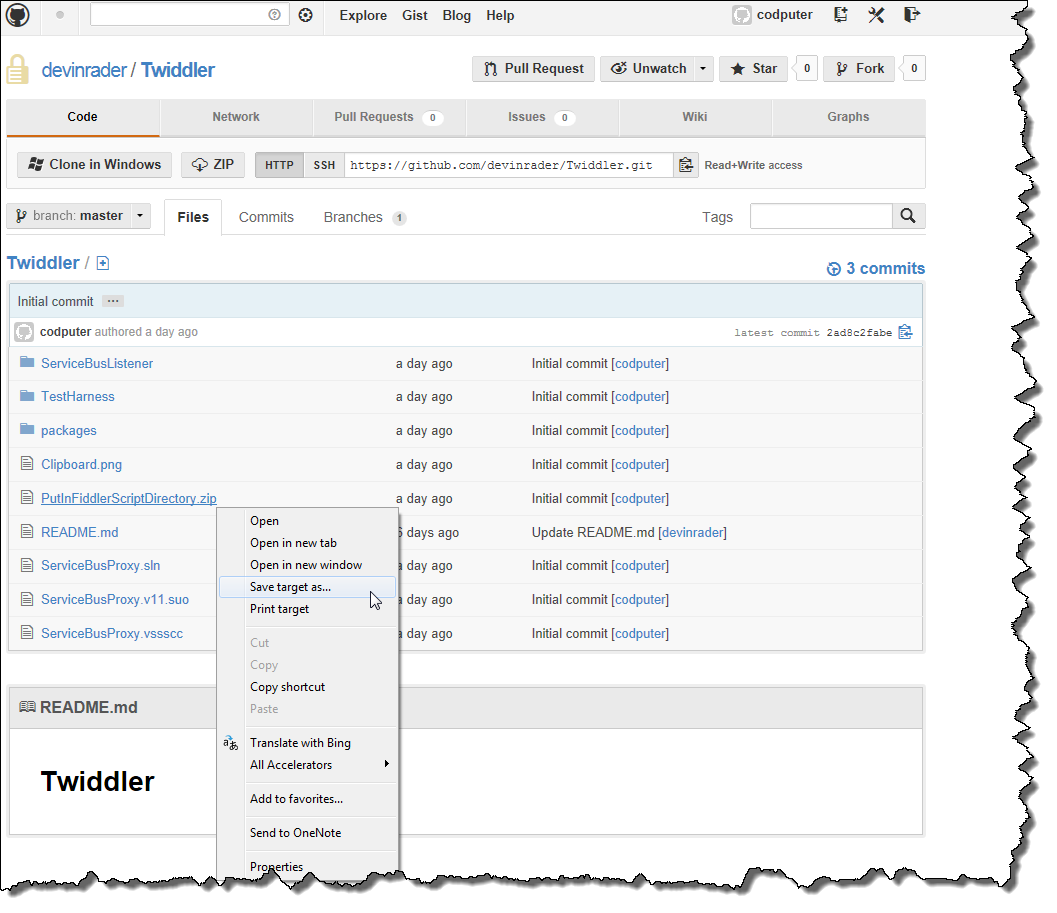
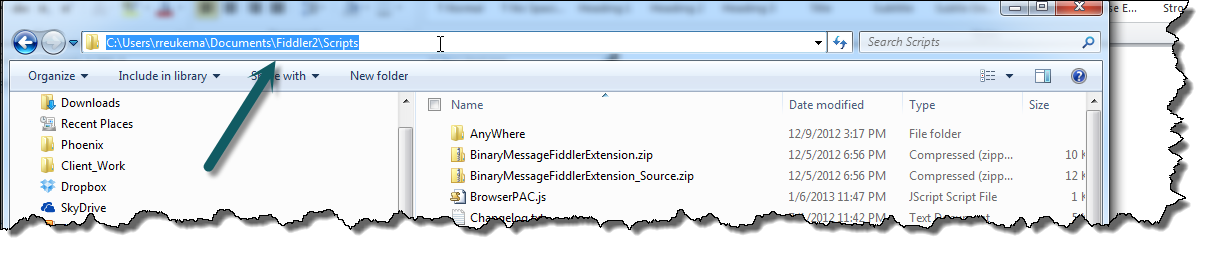
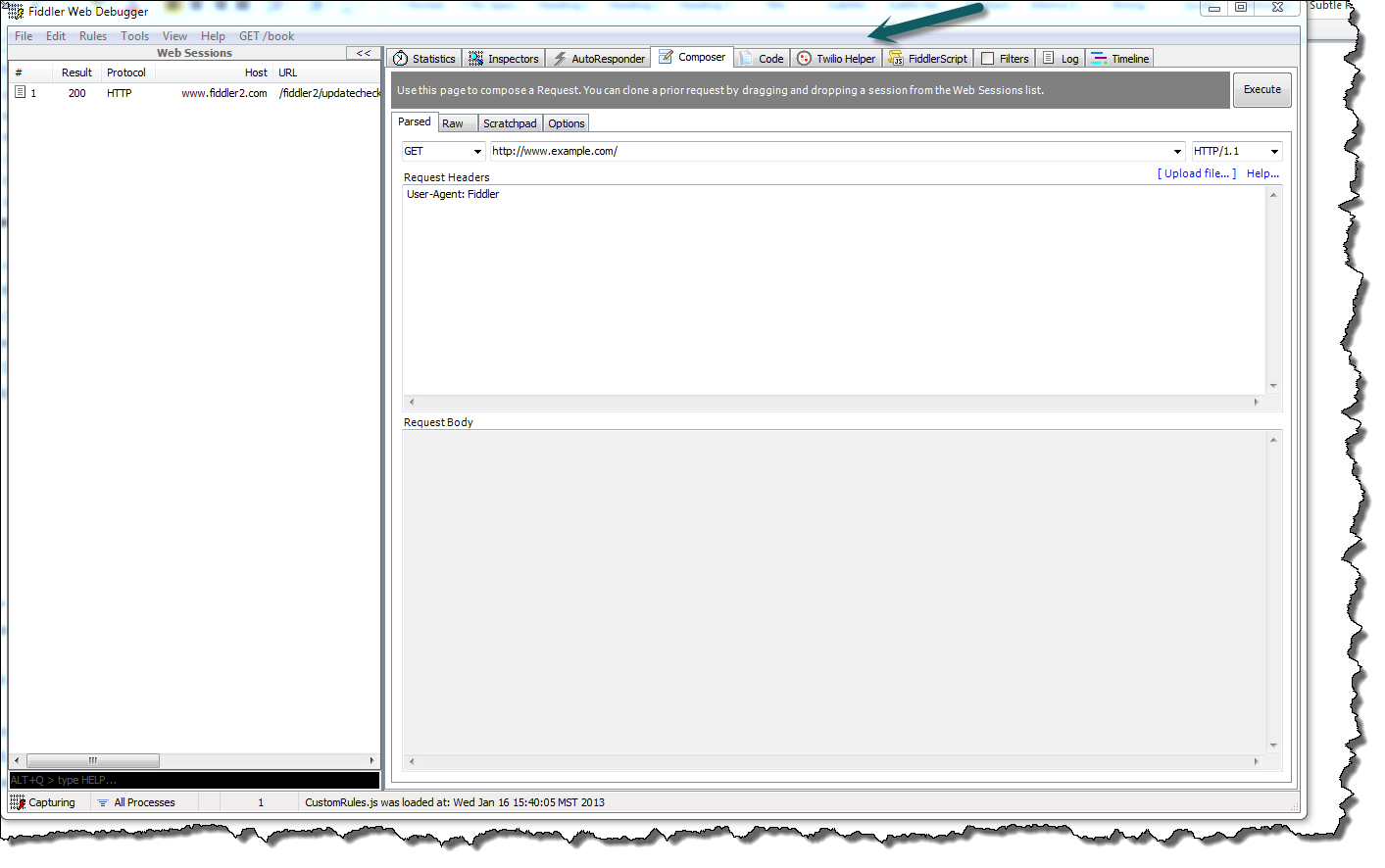
# Install Fiddler

NOTE: If you have Fiddler already installed, make sure **it’s Fiddler 4.0 (Beta)** application. Twiddler will NOT work with versions previous to version 4.0

1. Navigate To: <http://www.fiddler2.com/fiddler2/version.asp>
2. Download Fiddler 4.0  
   
3. Unblock the ZIP file
   1. Right click the filename, and select Properties  
      
   2. Now double click on the file, and accept all defaults.
   3. Navigate to the Scripts Folder  
      After installing Fiddler, a folder called Scripts will be created in your Documents folder under your profile name (mine is rreukema). Keep this path Open as you will require to navigate here shortly  
        
      NOTE: Do NOT start Fiddler quite yet… the extension code needs to be copied to this folder before Fiddler is started.

# Twiddler

## Installing Twiddler

1. Navigate to <https://github.com/devinrader/Twiddler>
2. Download Twiddler  
   Twiddler is available with complete source code, or only the Binary files necessary to run Twiddler.
   1. Binary Files Only
      1. Download the file “PutInFiddlerScriptsFolder.zip”as per below:  
         
      2. Save the file to the Fiddler Scripts folder as identified above (and duplicated here)  
         
      3. Unblock the File
         1. Right Click the file, and select Properties
         2. Click on the Unblock button
      4. UnZip the File
         1. Double click the file, and allow it to expand ***within the Scripts Folders***
      5. Start Fiddler  
         The extension will appear as a new tab (see section on “Configuring Twiddler” for detailed instructions on configuring and how to use Twiddler.  
           
         NOTE: If you cannot see this Tab, please see the section on Fiddler trouble shooting later in this document
   2. Complete Source
      1. Download all Files
      2. Ensure the following SDK’s are installed:
         1. Azure SDK
         2. Twilio API
         3. Microsoft.ServiceBus
         4. Fiddler.exe   
            Create a reference to Fiddler.exe via the Browse functionality of “Add Reference”
   3. Post Build command  
      The Project has a post build command that copies the Twiddler code from VS directory, to the default Fiddler script folder. NOTE: If Fiddler is running, the copy command will error out

## Configuring Twiddler

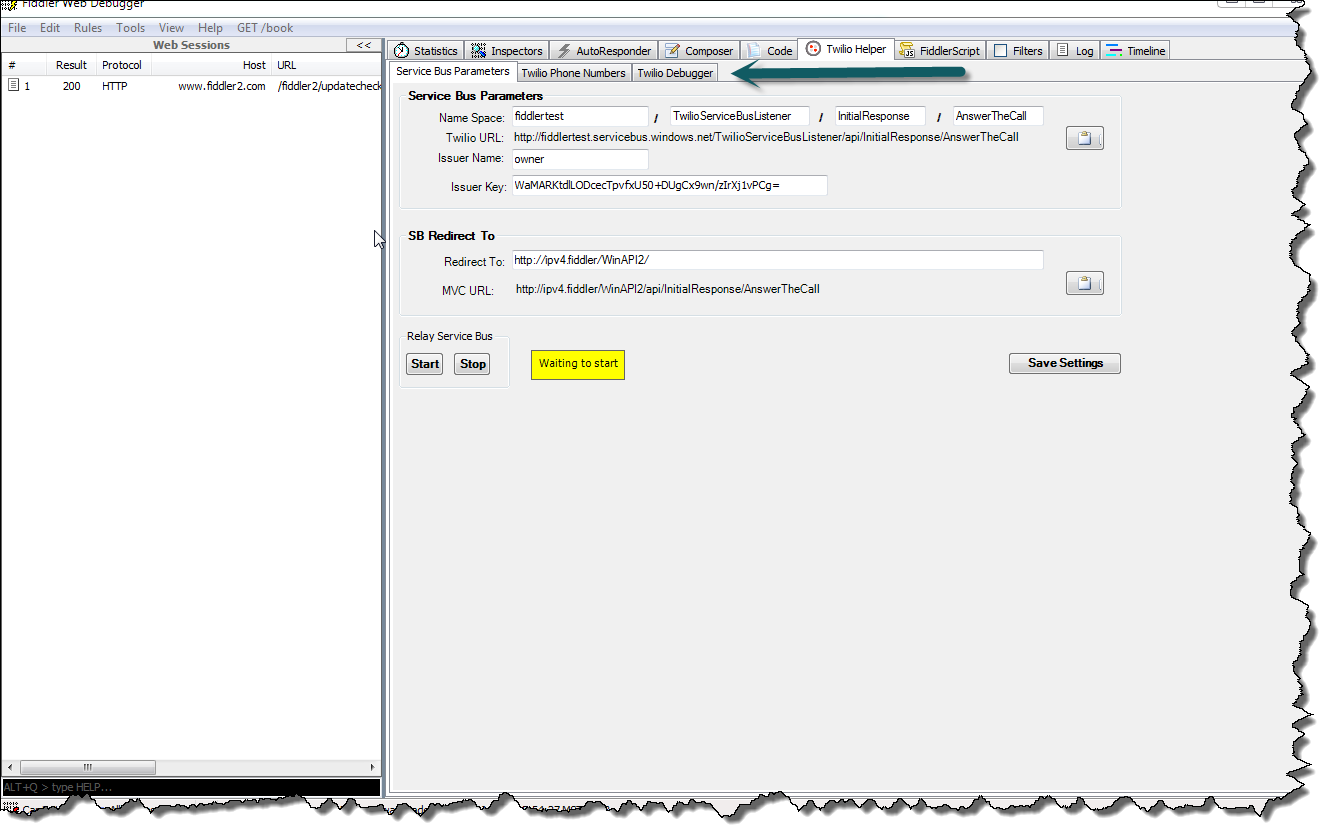
Twiddler has three tabs:

* Service Bus Parameters  
  Used to authorize Twiddler to set-up a Service Bus relay on your Azure account
* Twilio Phone Numbers  
  Used to authorize Twiddler to setup Twilio parameters on your Twilio Account.
* Twilio Debugger  
  Retrieves Twilio Notifications – used for debugging Twilio applications when something goes wrong on the Twilio side of the exchange.

Twiddler was configured this way to provide as much functionality via Twiddler, and minimize the effort to build and debug Twilio applications. Without Twiddler functionality, I was always jumping between numerous web browsers to debug and develop the application. Twiddler minimizes these interactions so that development can occur between your development environment (Visual Studio), and Twiddler.

# Service Bus Configuration

The Service Bus tab assists in the creation of the first URL that is used to enter the Twilio Voice URL



## Field Definitions

As noted during the creation of the Azure Account, please enter the following fields:

1. Name Space Row
   1. Name Space (first field)  
      Type in the Service Bus name space name as it appears in the portal (without the “.servicebus.windows.net” suffix)
   2. EndPoint Name/Service Path Name (second field)  
      Within in a Service Bus Namespace, you must specify an EndPoint Name, and it can be anything you like (no special characters). This field is used to create multiple endpoints within a single namespace. Within Twidder, we only require one.
   3. Controller (third field)  
      This is the name of the MVC controller that the Twilio request will be directed to. Please enter the name of the controller, with the “controller” suffix removed. The URL that we are creating for use in Twilio is being created directly below the current line.
   4. Action (fourth field)  
      This is the name of the MVC controller that the Twilio request will be directed to. This is the Action that will be called when the Twilio phone number is called.l

Directly below these 4 fields, an URL is being created. This URL is the URL that must be entered into Twilio as the Voice URL of your application (could also be the SMS URL, depending on the type of Application you are creating). For your convenience, clicking the clipboard button to the right will put this line into the clipboard, make it ready to paste into the Twilio parameters side of things.

1. Issuer Name  
   Enter the issuer name as it appears on the credentials popup (default to owner). When Security becomes a concern, this name should be change when a new issuer name is created with minimal security rights.
2. Issuer Key  
   As noted during the creation of the Azure account, use the “Access Key” button near the bottom of the Windows Azure portal.
3. Redirect To:   
   Type in the full root of the MVC or WebAPI host that is listening for requests. This will typically be <Http://localhost/XXXXXX> where XXXXX is the name of the web project in your solution. NOTE: if you enter localhost, Twiddler will replace it with “ip4.fiddler”. This is done to ensure Fiddler directs the request correctly to the localhost and avoids the issue of the loopback adapter.

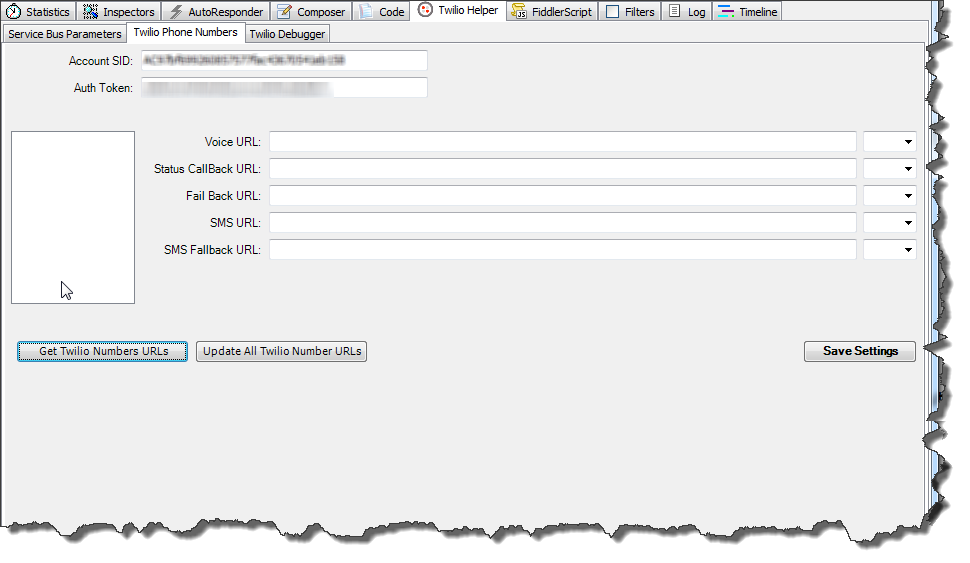
## Command Buttons

* Start  
  Providing the Issuer Name is correct, as well as the Issuer Key – Twiddler will establish a Service Bus endpoint that is hosted within Azure. This is a public endpoint that will route all web requests directly to your local workstation. If the endpoint is established successfully, a message will appear stating “Service Bus listener has started successfully”. Should an error occur, a message box will appear directly underneath the Start button.
* End  
  Simply stops the service, and the public endpoint is no longer accessible (and will not continue to accumulate “relay hours”)
* Save Settings  
  Saves the definition of all fields entered. When Fiddler/Twiddler is re-started, the fields will be populated from the last time this button was pressed.

## Usage

After the “Start” button is pressed, a public endpoint is created using the NameSpace name, as well as the EndPoint name. The complete “Twilio URL” should be used to configure the “Voice URL” of the Twilio account (see the Twilio Phone Numbers tab for assistance in setting this URL within Twilio). After the Service Bus Relay has been started, your local workstation is now listening on that endpoint for any requests that may be issued to that endpoint. Upon receiving that request, Twiddler will redirect the request (doing an URL rewrite), and route it directly to the host that was specified in “Redirect To” ( the NameSpace name, and the EndPoint name are replaced with the “Redirect To” host that was specified). NOTE: The URL in “Redirect To” need not be sent to a local web service, but could be redirected to ANY web service that is properly configured. For example, after my solution was deployed to Azure, I choose to have Twilio send the request to the Service Bus endpoint, where I then redirected the request to my Azure account.

# Twilio Configuration



## Field Definitions

* Account SID  
  As previously noted during the Twilio configuration, copy and paste in the Account SID from the Twilio Account that was created.
* Auth Token  
  As previously noted during the Twilio configuration, copy and paste the Auth Token from the Twilio Account that was created. NOTE: Click on the lock Icon first to unlock the contents, which will reveal the code. Copy this field only after it has been unlocked.
* Voice URL  
  Twilio will use this URL when someone dials the phone number which is selected on the left.
* Status Call Back URL  
  An optional URL that Twilio will use when the user hangs up the phone.
* Fall Back URL  
  An optional URL that Twilio will use should the Voice URL have a failure of some type. If not specified, Twilio has a default message that will be relayed.
* SMS URL  
  An optional URL that Twilio will use should some send a text to the phone number which is selected on the left.
* SMS Fallback URL  
  Same as the Voice Fallback URL, but only used if the Twilio application has a failure processing the SMS URL.

## Command Buttons

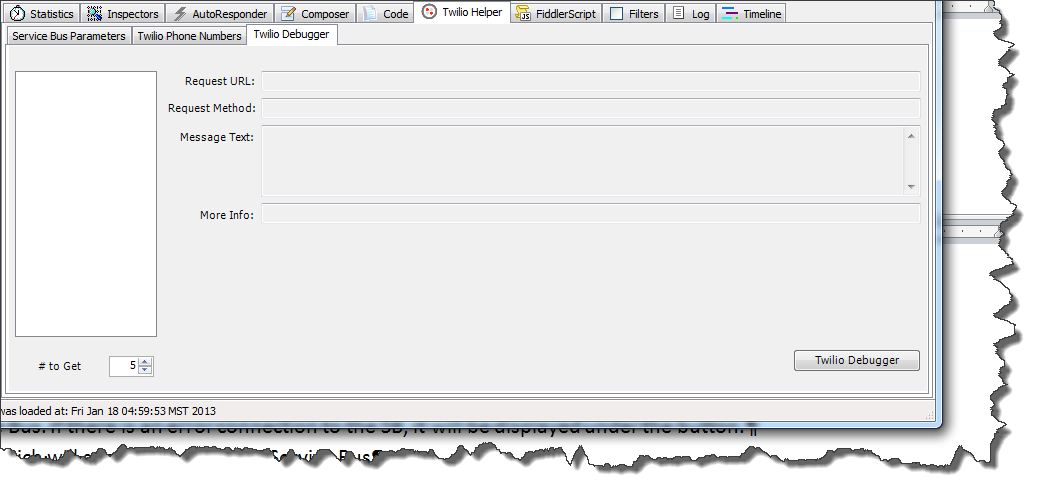
* Get Twilio Number URLs  
  During the Twilio configuration, a Twilio number was purchased. If you click on the button, all phone numbers purchased from Twilio will be displayed in the list box. The fields on the right will be populated with URLs that were previously saved. Initially these fields will be blank.
* Update All Twilio Number URLs  
  This button will update all the Twilio fields for each phone number listed in the list box.
* Save Settings  
  Saves the Account SID, and Auth Token fields. When Fiddler/Twiddler is re-started, the fields will be populated from the last time this button was pressed.

## Usage

1. Click on the “Get Twilio Number URLs” to retrieve all purchased phone numbers on the Twilio account specified (Account SID, Auth Token combination), along with the URLs that may be associated with each phone number.
2. Select the phone number within the list box.
3. From the Service Bus parameter tab of Twiddler, copy the “Twilio URL” to the clipboard (click on the clipboard button on the right).
4. Paste that URL into the Voice URL (or perhaps the SMS URL depending on your application).
5. Click on the “Update All Twilio Number URLs”. This will take the current URL information for each phone number, and transfer this information to Twilio. Should errors occur, an error message will appear directly under the buttons.

# Twilio Debugger

When an error occurs during a Twilio exchange, Twilio will capture the error on their side, and record it. This tab allows for the retrieval of those errors without having to bring up the Twilio web site in another browser.



## Field Definitions

* Request URL  
  The URL Twilio used in which the error occurred.
* Request Method  
  Twilio can use either a GET or POST method. This field indicated which method was used.
* Message Text  
  The error message as it was retrieved from Twilio.
* More Info  
  Any type of information that was captured from the Twilio request that may prove helpful in debugging the application
* # to Get  
  Simply the number of errors to retrieve – errors are retrieved in reverse chronological order.

## Command Buttons

* Twilio Debugger  
  Uses the Account SID/Auth Token to retrieve the number of errors specified on the left. Date and time is listed on the left, and detail information of the error is listed on the right.

## Usage

1. Account SID/Auth Token  
   Ensure that these fields have been specified under the Twilio Debugger tab.
2. Click the “Twilio Debugger” command button to retrieve errors from the Twilio Account.

# Trouble Shooting

## Fiddler

As explained [here](http://www.fiddler2.com/Fiddler/dev/IFiddlerExtension.asp), Fiddler by default ignores all errors within extensions and silently fails should an error occur. If you want a little more info from Fiddler, you have to set some preferences in order to display the error messages. In the “Debugging your Extensions” section, it indicates that you need to set the “fiddler.debug.extensions.showerrors” preference to true, as well as “fiddler.debug.extensions.verbose” preference to true. In order to do this, type the following commands into the QuickExec box (see [here](http://www.fiddler2.com/Fiddler/help/ui.asp) for UI help).

Enter the following commands:

Set Pref fiddler.debug.extensions.showerrors true  
Set Pref fiddler.debug.extensions.verbose true

