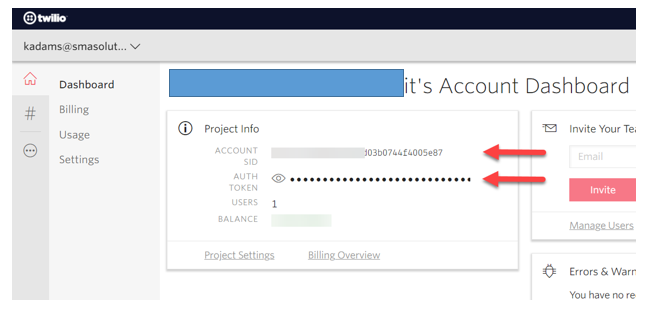
OpCon – Phone Call Notifications via Twilio

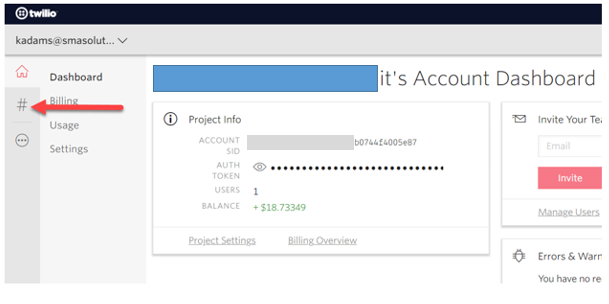
# Twilio Configuration Values

After creating a Twilio account, you can log into your Twilio console. The Account SID and Authorization token are shown on the home page.

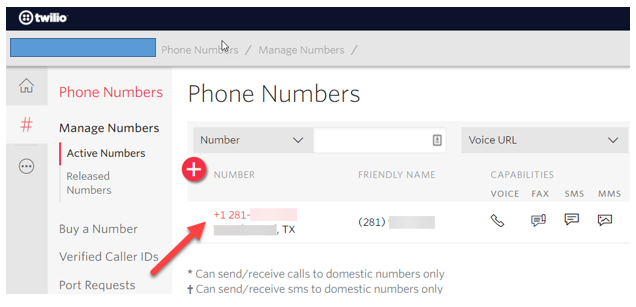


Record these values!

Next, click on the number sign (#) to see the Authorized phone number(s):



This will display a screen like this one:



Record this telephone number. It is the authorized telephone number for this account and is needed for the SMATwilioConnector configuration.

# Twilio Configuration Values

Powershell script which can be used to initiate the Twilio call.

<#

InitiateTwilioCall.ps1

Version History:

2018/05/10: Initial creation with basic functionality and

logging.

#>

param(

[string]$TwilioAccountSID,

[string]$TwilioAuthToken,

[string]$TwilioNumber,

[string]$PhoneNumber,

[string]$Message

)

try

{

Add-Type -AssemblyName System.Web

Write-Host "Target Phone Number: $PhoneNumber`n"

Write-Host "Voice Message: $Message`n"

$Message = [System.Web.HttpUtility]::UrlEncode($Message)

$TwiMLUrl = "http://twimlets.com/message?Message%5b0%5d=$Message"

Write-Host "TwiML URL: $TwimlUrl`n"

# Twilio API endpoint and POST params

$url = "https://api.twilio.com/2010-04-01/Accounts/$TwilioAccountSID/Calls.json"

$params = @{ To = "+$PhoneNumber"; From = "+$TwilioNumber"; Url = $TwiMLUrl }

# Create a credential object for HTTP basic auth

$p = $TwilioAuthToken | ConvertTo-SecureString -asPlainText -Force

$credential = New-Object System.Management.Automation.PSCredential($TwilioAccountSID, $p)

# Make API request, selecting JSON properties from response

$result = Invoke-WebRequest $url -Method Post -Credential $credential -Body $params -UseBasicParsing | ConvertFrom-Json

Write-Host "Response: `n"

$result | Format-List

}

catch

{

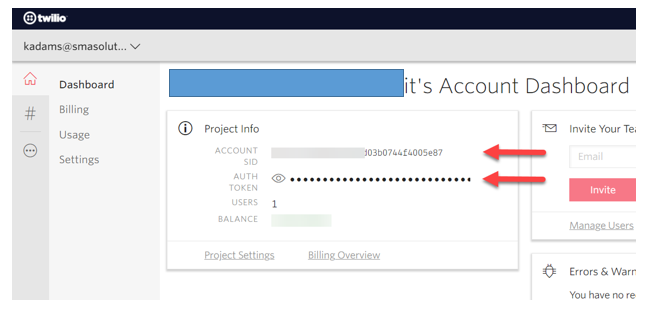
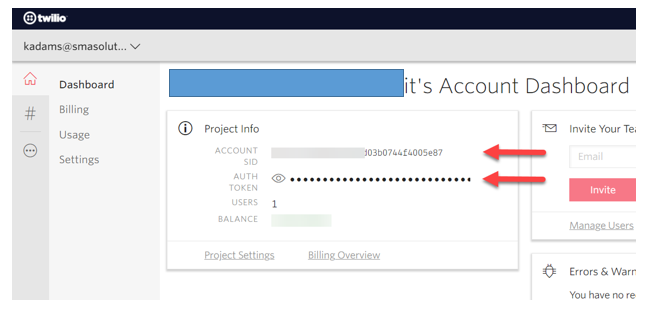
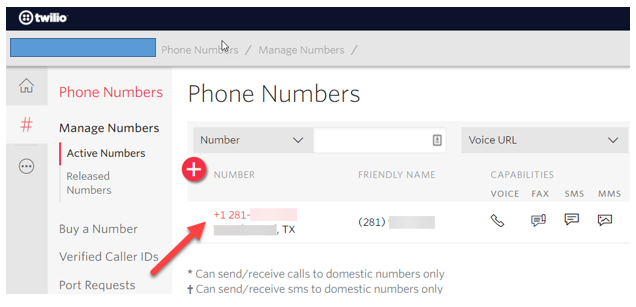
$\_.Exception | Format-List

Exit 1

}

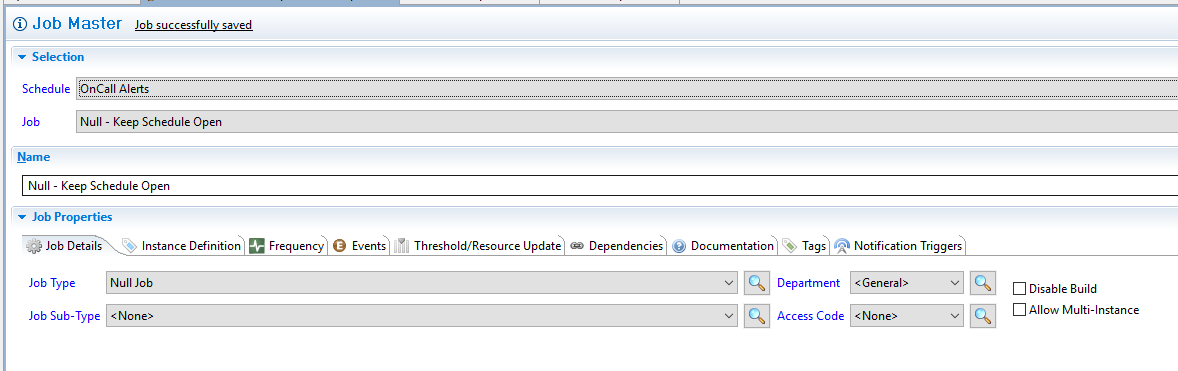
# Command Line Parameters

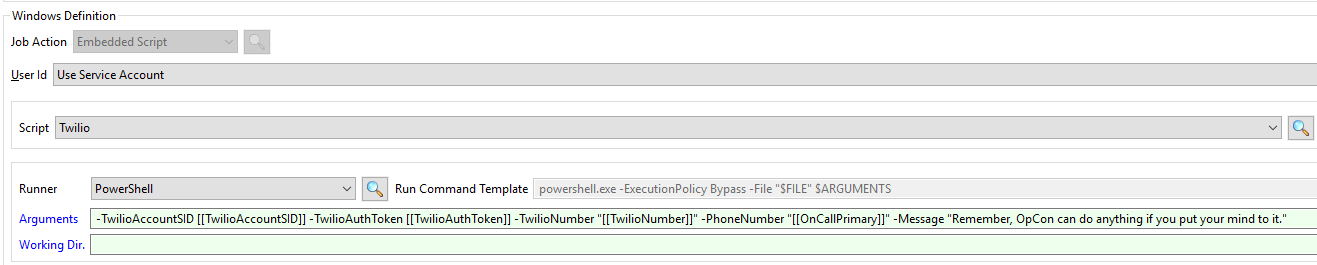
The following Command Line Parameters are required in the OpCon Job.

* **TwilioAccountSID** = This is the Twilio Account SID you will be using to make the call.  
  
* **TwilioAuthToken** = This is the Authentication Token granting the Account SID access to Twilio through its Rest API.  
  
* **TwilioNumber** = This is the number which is registerd to the Twilio Account to make phone calls. This will be the incoming number your on-call staff will see.  
  
* **PhoneNumber** = This is the number which will be called.
* **Message** = This is the message which will be given during the call.

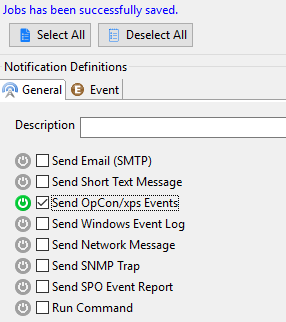
# OpCon Setup

Create an OnCall Schedule adding at minimum the following two Jobs. Then add the Notification Events in Notification Manager.

* **Job 1**: Is a **Null Job** which runs every day at **24:00**. The purpose of this Job is to keep the Schedule open throughout the day allowing Twilio Jobs to be added when a critical failure occurs.  
  
* **Job 2**: Is a **Windows Job** which has an **OnRequest** Frequency. This Job will contain the Command Line to make a Twilio Call to the 1st Level of On-Call support.



-TwilioAccountSID [[TwilioAccountSID]] -TwilioAuthToken [[TwilioAuthToken]] -TwilioNumber "[[TwilioNumber]]" -PhoneNumber "[[OnCallPrimary]]" -Message "Remember, OpCon can do anything if you put your mind to it."

* -TwilioAccountSID [[TwilioAccountSID]]
* -TwilioAuthToken [[TwilioAuthToken]]
* -TwilioNumber "[[TwilioNumber]]"
* -PhoneNumber "[[OnCallPrimary]]"
* -Message "Remember, OpCon can do anything if you put your mind to it."
* **Notification Manager**: Setup a **Group** with the appropriate **Jobs** assigned and appropriate triggers. Select the “**Send OpCon/xps Events**” option.  
  
  + The event should be a $JOB:ADD event with the initial Twilio OnCall Job.

# Escalating Calls

To setup escalating calls you can add more Jobs for each escalation tier. You can then set each Job’s Frequency to loop for X number of runs with Y minutes between. Then do either a $THRESHOLD:SET incrementing a Threshold +1 or a $PROPERTY:SET using a Property Expression updating a Schedule Instance Property after each loop Finishes Ok. Then setup an event based on an expression which would add the Job escalating the phone call once the Threshold or Property hit a value of X.

## Theshold setup

Job Completion Event Incrementing the Threshold’s value:

$THREHOLD:SET,Escalation,+1

Expression Event triggering the $JOB:ADD for the next tier:

[[TH.Escalation]] == 6

## Schedule Instance Property setup

Job Completion Event Incrementing the Schedule Instance Property’s value:

$PROPERTY:ADD,SI.Escalation.[[$DATE]].”[[$SCHEDULE NAME]]”,[[=ToInt([[SI.Escalation]])+ToInt(1)]]

Expression Event triggering the $JOB:ADD for the next tier:

[[SI.Escalation]] == 6