

# Automation, Notification and Jubilation: Incorporating Notifications into Geoprocessing Workflows

---

SOUTH CENTRAL ARC USER GROUP

APRIL 16, 2015



# Introduction

---

Geoprocessing scripts boost productivity, improve service, and promote data integrity.

Scripts in production are often ignored until something goes wrong.

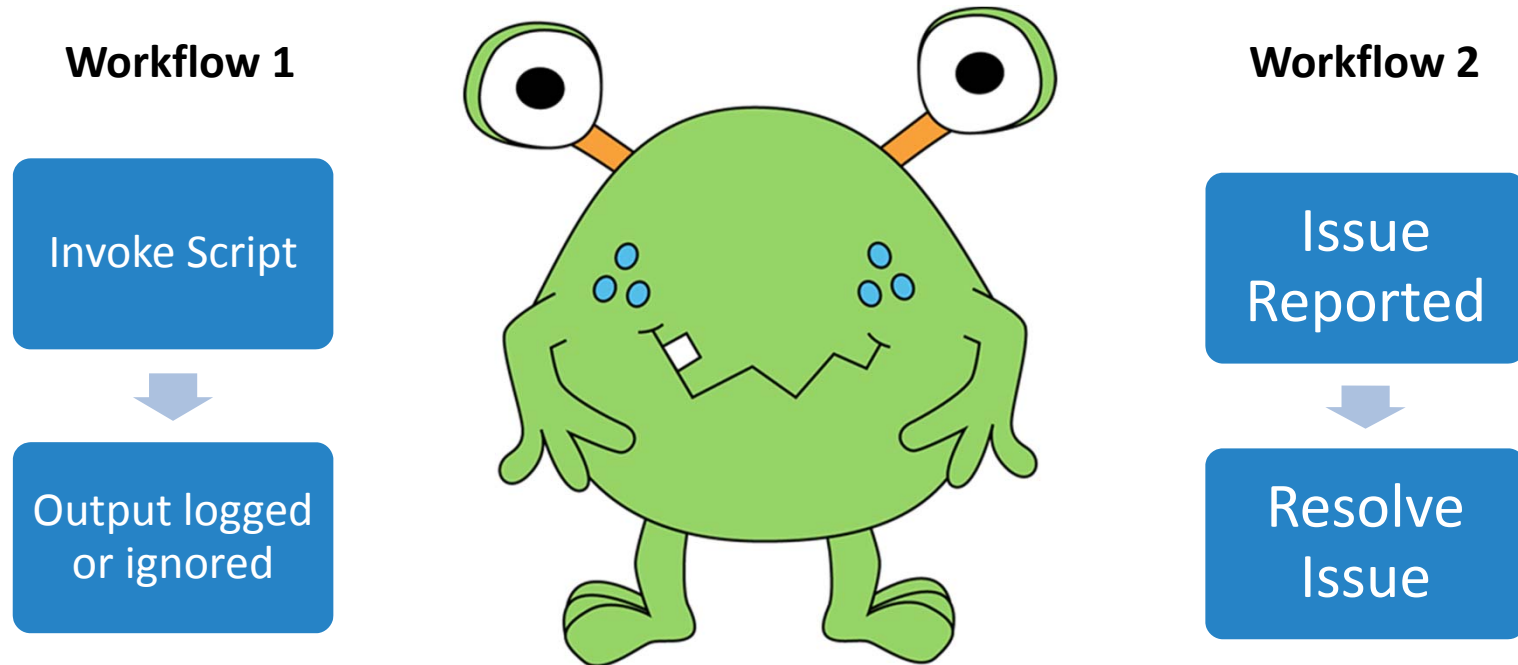
Script failure may lead to stale data, service disruptions, and customer dissatisfaction.



# The Problem

---

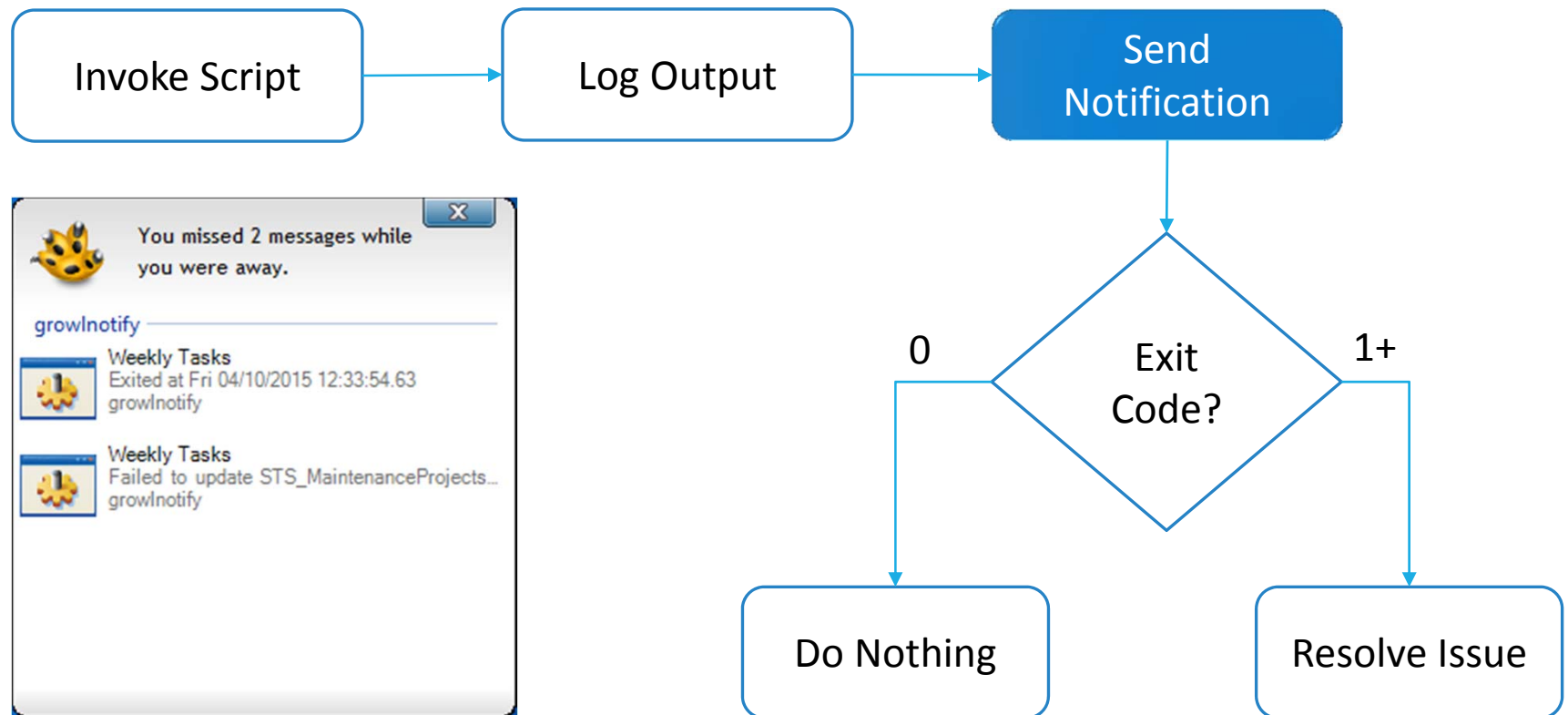
## Undetected Geoprocessing Script Failure



Source: <http://www.mycutegraphics.com>

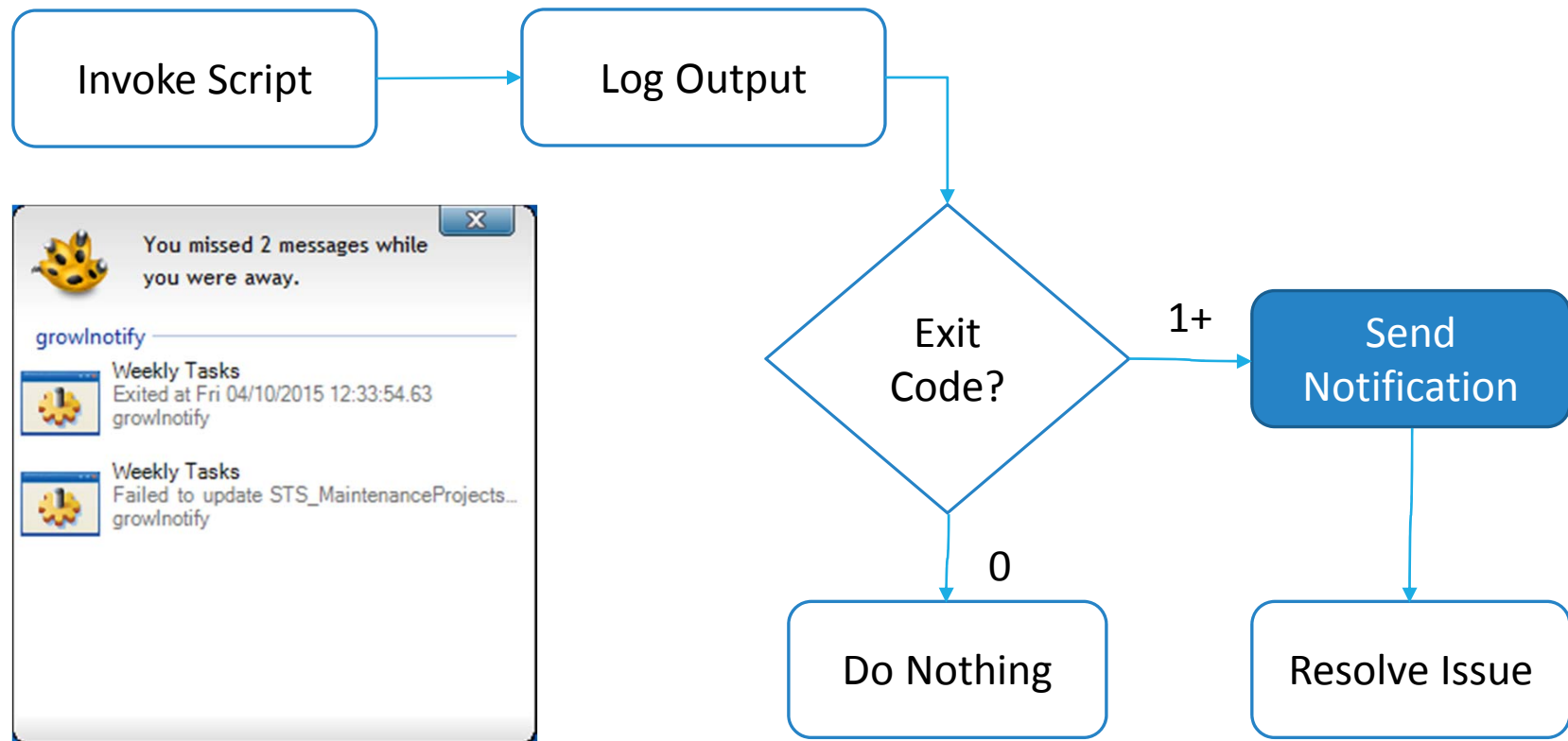
# The Solution

## Incorporate Notifications into Geoprocessing Workflows



# The Solution

## Incorporate Notifications into Geoprocessing Workflows



# The Solution

---

Geoprocessing scripts return an exit code where zero indicates successful completion and nonzero value is considered abnormal termination.

The exit code can be tested in a batch file to send a notification upon successful completion or abnormal termination (failure).

As a consequence, the notification method must have a command line interface.

The recommended approach is to keep the geoprocessing script and notification script/utility separate.




# Notification Methods

---

## Purpose

- Inform the recipient whether an action is needed or not needed.

## Communication Process

- **Sender:** The batch command that invokes the Geoprocessing script.
  - **Recipient:** One or more persons that need to take action or no action based upon the message.
  - **Message:** The message typically identifies the script name, exit status, and timestamp. The message may include log messages and attachments.
  - **Channel:** The channel is typically email or push notification.
- 

# Email

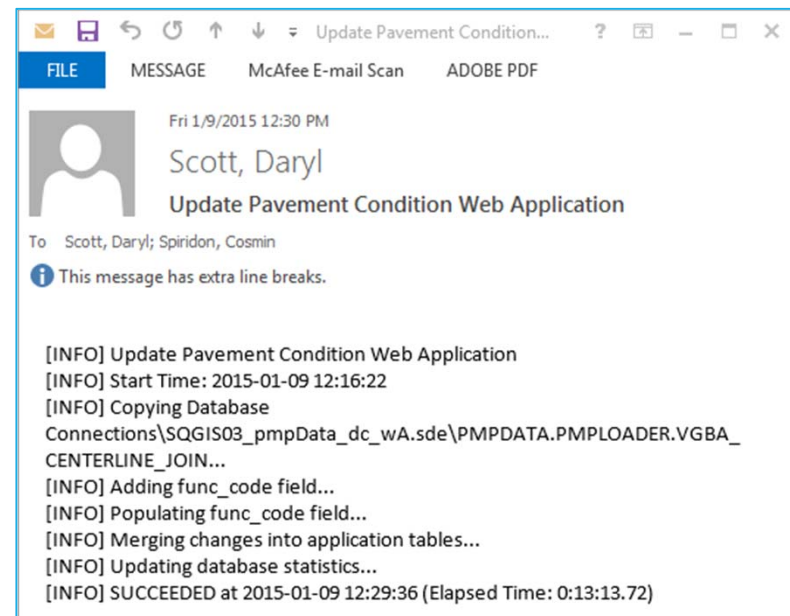
---

Send message to one or more email addresses

Permits lengthy messages and attachments

Command Line options:

- Automate Microsoft Outlook using Python and Pythonwin
- Automate sending message through SMTP
- 3<sup>rd</sup> Party utilities





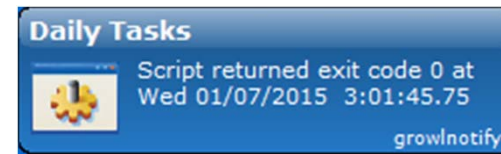
# Growl for Windows

Provides standard notification system for one or more applications

Forward notifications to other computers and email addresses.

Supports popular programming languages such as .Net, Python, and JavaScript

<http://www.growlforwindows.com/gfw/>



# GrowlNotify

---

## DOWNLOAD

[Download growlnotify command line script](#)

## NAME

`growlnotify` -- Send a Growl notification to a local or remote host

## SYNOPSIS

```
growlnotify [/t:title] [/id:id] [/s:sticky] [/p:priority] [/i:icon] [/c:coalescingid]
            [/a:application] [/ai:appicon] [/r:types] [/n:type]
            [/cu:callbackurl]
            [/host:host] [/port:port]
            [/pass:password] [/enc:algorithm] [/hash:algorithm]
            [/silent:suppressoutput]
            messagetext
```

## DESCRIPTION

Only `messagetext` is required. All other switches will use default values.

<code>/t:title</code>	The notification title. Use <code>\n</code> to specify a line break. Use <code>\\n</code> for a literal <code>'\n'</code> . Default: "growlnotify"
<code>/id:id</code>	The notification id. Default: ""
<code>/s:sticky</code>	Indicates if the notification should be sticky. Valid values: true false Default: false

Source: <http://www.growlforwindows.com/gfw/help/growlnotify.aspx>



# Pushbullet

---

Push messages, links, and files to devices or email addresses

Supports iPhone, Android, Windows, and browser add-ons

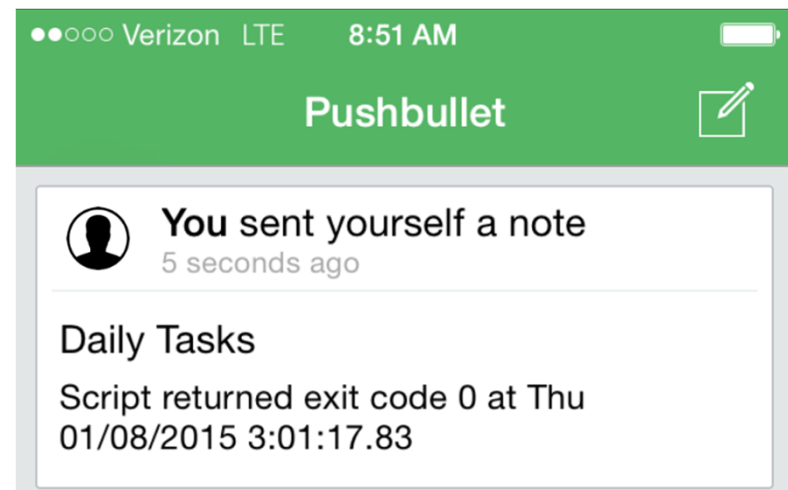
Easy to use developer API

Note: Requires Google or Facebook account.

Python library available on GitHub

<https://github.com/randomchars/pushbullet.py>

<https://www.pushbullet.com/>

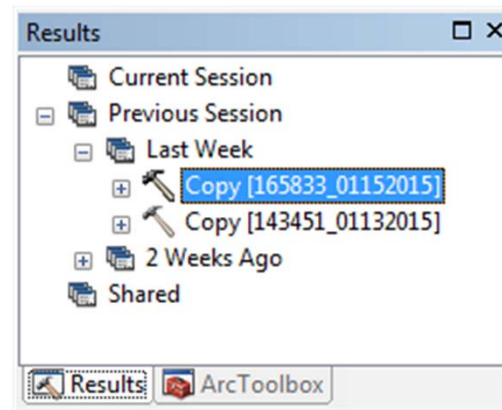


## Step 1—Create Geoprocessing Script

A Geoprocessing Model can be exported to a Python script.

In the Geoprocessing Results window, a command with its parameters can be copied to the clipboard using the “Copy As Python Snippet” command.

Pay attention to Geoprocessing environment such as workspace, outputCoordinateSystem, and overwriteOutput. The environment settings are set by calling application or script.



Established Workflow



Create Geoprocessing Script

```
import arcpy
in_data = "C:/PBW/GIS/benchmark.gdb"
out_data = "C:/PBW/GIS/benchmark_backup.gdb"
arcpy.env.overwriteOutput = True
arcpy.management.Copy(in_data, out_data)
print arcpy.GetMessages()
```

Figure 1: Backup file geodatabase script

## Step 2— Setup Logging

Logs are an important tool for monitoring and troubleshooting problems.

If exception handling is present, the exit code should be explicitly set using the `sys.exit` method.

### Method 1: Redirect messages to log file

**REM** Append both standard output and standard error of command to file

```
benchmark_backup.py >> benchmark_backup.log 2>&1
```

Figure 2: Execute script with output redirection

### Method 2: Incorporate logging into geoprocessing

```
import arcpy
import sys

try:
    status = 0
    in_data = "C:/PBW/GIS/benchmark.gdb"
    out_data = "C:/PBW/GIS/benchmark_backup.gdb"
    log_file = "benchmark_backup.log"
    arcpy.env.overwriteOutput = True
    arcpy.management.Copy(in_data, out_data)
except:
    status = 1
finally:
    with open(log_file, 'a') as f:
        f.write('\r\n' + arcpy.GetMessages() + '\r\n')

sys.exit(status)
```

Figure 3: Backup file geodatabase script with logging

## Step 3— Setup Notification

The purpose of notifications is to inform the recipient whether an action is needed or not needed.

Does the recipient need to troubleshoot an issue?

Does the batch need to be executed manually?

Use the `%ERRORLEVEL%` variable to test exit status of the most recent command

```
benchmark_backup.py >> benchmark_backup.log 2>&1  
  
growlnotify.com /s:true /t:"Backup Task" "Script  
returned exit code %ERRORLEVEL% at %DATE% %TIME%"
```

Figure 4: Send notification regardless of exit status

```
benchmark_backup.py >> benchmark_backup.log 2>&1  
  
if %ERRORLEVEL% neq 0 growlnotify.com /s:true  
/t:"Backup Task" "Failed to backup geodatabase at  
%DATE% %TIME%"
```

Figure 5: Send notification only after abnormal termination

```
benchmark_backup.py >> benchmark_backup.log 2>&1  
  
if %ERRORLEVEL% equ 0 growlnotify.com /s:true  
/t:"Backup Task" "Backup geodatabase succeeded at  
%DATE% %TIME%"
```

Figure 6: Send notification upon normal termination

## Step 3— Setup Notification

Multiple notification calls require care since the %ERRORLEVEL% may correspond with the most recent notification call instead of the Geoprocessing script.

### Use the %ERRORLEVEL% variable (Continued)

How to avoid unnecessary notifications

- Create a batch file that executes multiple geoprocessing tasks, e.g. daily\_tasks.bat, weekly\_tasks.bat, or backup.bat
- Send failure notification only if an individual task fails
- Send notification upon batch completion

```
geoprocessing_task1.py geoprocessing_task1.log  
if %ERRORLEVEL% neq 0 growlnotify.com ...  
  
geoprocessing_task2.py geoprocessing_task2.log  
if %ERRORLEVEL% neq 0 growlnotify.com ...  
  
geoprocessing_task3.py geoprocessing_task3.log  
if %ERRORLEVEL% neq 0 growlnotify.com ...  
  
growlnotify.com ...
```

Figure 7: Send notification upon task failure and upon batch completion

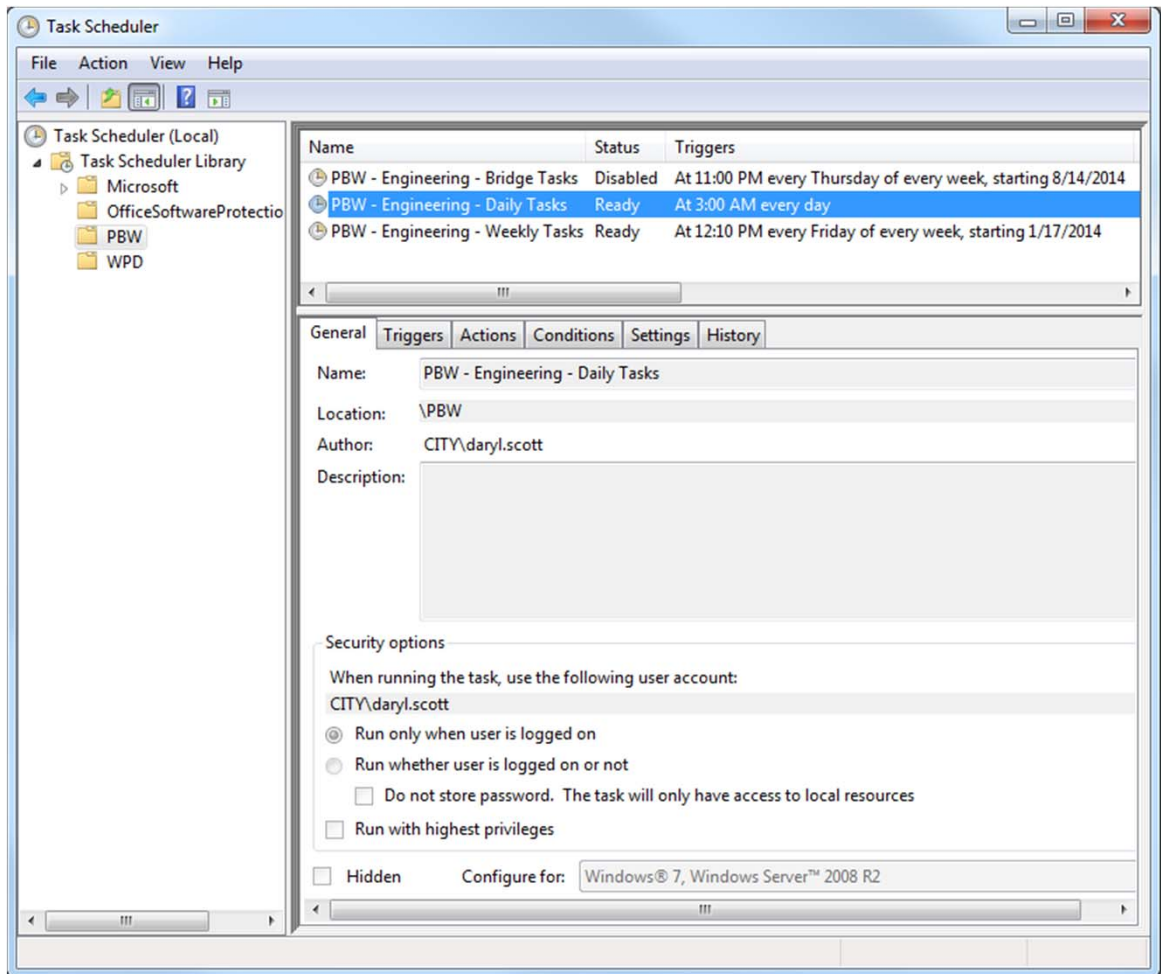
# Step 4— Schedule Automation Script Execution

Create folder for user-defined tasks

Batch file may need to be wrapped in VBScript or invoked using cmd.exe.

Alternative: Freebyte Task Scheduler

## Windows Task Scheduler





# Summary

---

What? Incorporate notifications into geoprocessing workflows

Why? To prevent undetected geoprocessing script failure

How?

1. Create geoprocessing script
2. Setup logging
3. Setup notification
4. Schedule batch file execution

# What about jubilation?

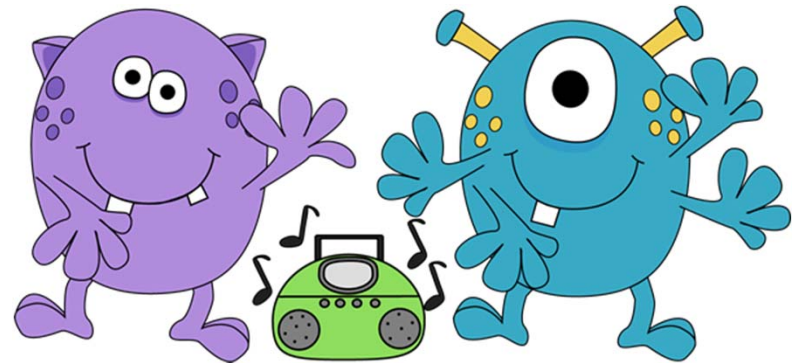
---

## Less

- Stale Data
- Service Disruptions
- Customer Dissatisfaction

## Benefits

- Personal Satisfaction
- Accolades
- Superior or Exceptional Performance Plan rating



Source: <http://www.mycutegraphics.com>

# Questions?

---

Daryl Scott

City of Dallas

[daryl.scott@dallascityhall.com](mailto:daryl.scott@dallascityhall.com)

(214) 948-4672