Tableau – VizAlerts Installation Guide

Contents

[What is VizAlerts? 2](#_Toc446430753)

[What does it do? 2](#_Toc446430754)

[How does it work? 2](#_Toc446430755)

[Prerequisites 3](#_Toc446430756)

[Tableau Server 3](#_Toc446430757)

[Windows Host Machine 3](#_Toc446430758)

[SMTP (Mail) Server 4](#_Toc446430759)

[Setup 4](#_Toc446430760)

[Configure Tableau Server 4](#_Toc446430761)

[Trusted Tickets 4](#_Toc446430762)

[Repository Access 4](#_Toc446430763)

[Restart 4](#_Toc446430764)

[Custom Subscription Schedules 4](#_Toc446430765)

[Install Python & Required Modules 6](#_Toc446430766)

[Configure VizAlerts 7](#_Toc446430767)

[Testing 8](#_Toc446430768)

[Can VizAlerts Connect? Test 8](#_Toc446430769)

[Simple Alert Test 8](#_Toc446430770)

[Put VizAlerts Through Its Paces Test 9](#_Toc446430771)

[Final Steps 11](#_Toc446430772)

[Set up a Scheduled Task 11](#_Toc446430773)

[Helper Datasource 14](#_Toc446430774)

[FAQ 14](#_Toc446430775)

[Common Errors 15](#_Toc446430776)

[Getting VizAlerts Help 15](#_Toc446430777)

[Contributing to VizAlerts 15](#_Toc446430778)

[Appendix A 15](#_Toc446430779)

# What is VizAlerts?

VizAlerts is an automation platform intended to seamlessly integrate with Tableau Server. The idea behind it is that **anyone** should be able to easily build, share, and customize nearly **any** any email or SMS **automation** based on their own Tableau Server viz data. In the future, VizAlerts will be able to perform even more actions than these.

In its current form, VizAlerts exists simply as a Python script that is set up by a system administrator to run at frequent and regular intervals. All of the information it needs to enable data-driven email and SMS alerting is derived from the Tableau Server PostgreSQL repository, and the published views of the Tableau Server it is set to run against.

# What does it do?

VizAlerts has been designed to support many use cases:

* Sending notifications to subscribers when a condition has been met (or not!) like extract refresh failures, meeting or missing target thresholds, etc.
* Halting emails from being sent to the group unless the data is up to date (while the workbook owner does get an email that the data isn’t up to date).
* Notify data owners when data is corrupt in some way (extraneous values, too many Null values, too large a change, etc.)
* Sending a one line email notification that could be forwarded through an email-to-sms gateway such as ########@txt.att.net. (Or sending a multimedia notification!).
* Batch reporting to distribution lists of non-Tableau users, for example emailing a weekly operations report to front-line staff who aren’t on Tableau.
* Combining images and text into an HTML email for a more customized, professional look.
* Merge multiple views into a single consolidated PDF, for example sending a company overview and per-region detail.
* Blend views from separate workbooks in the same email, for example a view from the operations dashboard and a view from the finance dashboard.
* Bursting reports, for example sending to a manager a dashboard for each of her direct reports.
* Send SMS messages to escalate an issue to your support staff.
* Whatever combinations of the above you can imagine!

# How does it work?

While the details on how to *set up* alerts will be left in the User Guide, it’s important for Administrators to know how things are working behind the scenes.

The general flow of a single execution of the vizalerts program goes like this:

1. Connect to the PostgreSQL repository database of Tableau Server, and query it for a list of Views subscribed to on specially configured, disabled Schedules
2. Compare Schedule information to last runtime information stored in a local text file—based on this, determine which Views are due for alert testing
3. For each View found due for testing:
   1. Generate a Trusted Ticket on behalf of the Subscriber of the View
   2. Redeem the Trusted Ticket to export the CSV data for the View, impersonating the Subscriber
   3. If one or more rows are found in the CSV:
      1. For a “Simple Alert”, generate a new Trusted Ticket, export the PNG of the View, and email it to the Subscriber.
      2. For an “Advanced Alert”, iterate through each row of the CSV, sending emails or performing other actions as instructed by the data itself.

Prerequisites

## Tableau Server

The Tableau Server instance that you wish to run VizAlerts against must fulfill the following requirements:

* Must be v8.2.5 or higher
* The [readonly user](http://onlinehelp.tableau.com/current/server/en-us/adminview_postgres_access.htm) must be granted Repository database access
* [Subscriptions](http://onlinehelp.tableau.com/current/server/en-us/subscribe.htm) must be enabled
* The host you plan to run VizAlerts from must have its IP address listed as a [Trusted Host](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth_trustIP.htm)
* If it wasn’t already obvious, you need to be a System Administrator on Server to set all this up.

## Windows Host Machine

This is where VizAlerts will be run from, which means that this machine must be continually up and running for VizAlerts to function. This can be one of the Tableau Server hosts if desired, but it doesn’t have to be. It must have the following properties:

* Static IP address
* Always running
* Within same domain as Tableau Server
* You must have administrative rights to it
* Should **not** need to have much processing power as heavy work is offloaded to Tableau Server

## SMTP (Mail) Server

VizAlerts needs to point to a mail server to send email. This can simply be the same server you used when you set up Tableau Server for subscriptions. If your mail server is set up to support SSL encryption, that is ideal, but it’s not required.

# Setup

You’ve got everything you need, now let’s get this thing running!

## Configure Tableau Server

Making any of these configuration changes requires a restart of Tableau Server, so if this is being done on a live / production server, make sure to do this during a maintenance window.

### Trusted Tickets

VizAlerts uses [Trusted Authentication](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth.htm) to impersonate users and obtain access to Tableau Server views in CSV and PNG format. To grant it this access, run the following command at a command prompt on the Primary host of Tableau Server:

tabadmin set wgserver.trusted\_hosts <HOSTNAME OF VIZALERTS HOST>

### Repository Access

The Tableau Server repository database contains information VizAlerts needs to function. Grant it access by enabling the [readonly user](http://onlinehelp.tableau.com/current/server/en-us/adminview_postgres_access.htm):

tabadmin dbpass --username readonly <YOUR PASSWORD>

### Restart

Once you have finished the above steps, you must save the configuration and restart Tableau Server. When you’re ready to do this, run the following commands in the command prompt:

tabadmin configure

tabadmin restart

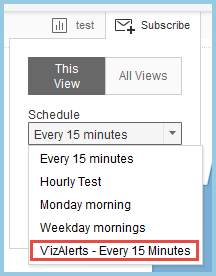
### Custom Subscription Schedules

A key component that allows VizAlerts to work in the intuitive way that it does is that users who wish to schedule an alert are able to subscribe to them on *disabled* Subscriptions schedules. These are schedules that you must create in Tableau Server, then manually disable so that no subscriptions are ever delivered for them. Since the data for who subscribed to what views *on* these specific schedules exists in the PostgreSQL repository, VizAlerts can use this information to tell itself when it is appropriate to test those views for an alert condition.

You can create as many schedules as you like, on whatever intervals you like. The important bit behind the schedules is the **naming convention** that you use, because this is how VizAlerts knows which schedules to consider “alert” schedules that it needs to pay attention to. I recommend naming them like this:

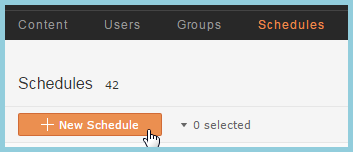
ѴizАlerts – [frequency]

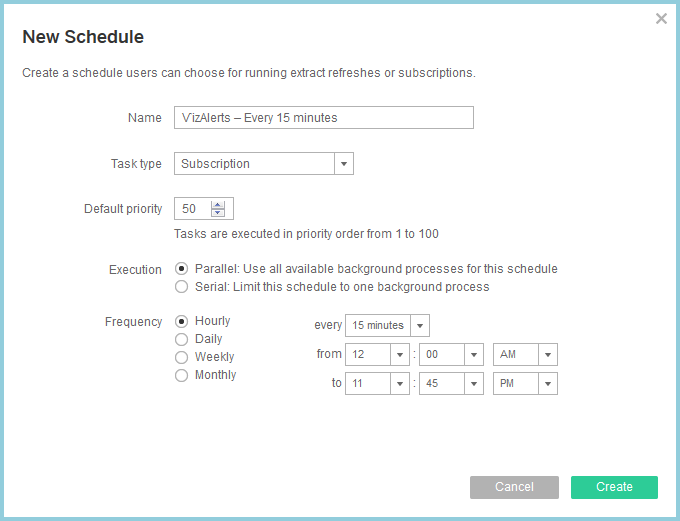
**Copy and paste that** when you create your schedules—the first letter is actually the Cyrillic letter Ѵ, which will cause your Alerts schedules to be sorted at the bottom of the list when someone goes to subscribe. This can help users avoid subscribing to them by mistake when they only mean to set up a standard subscription:

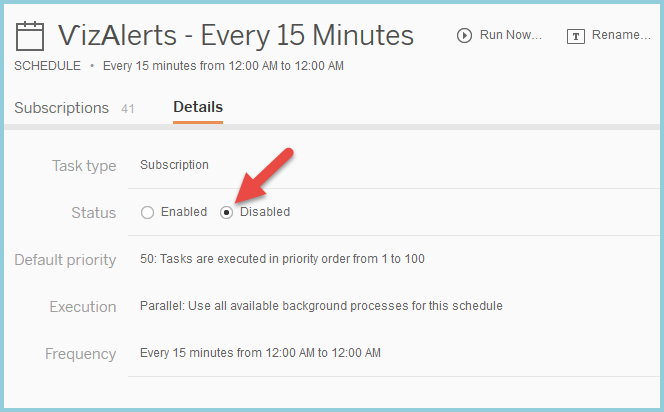


Note that you must have at least one **enabled** Subscriptions schedule for anyone to subscribe to a viz on Tableau Server, so if you have just enabled Subscriptions for the first time, you’ll also need to create a single non-Alert schedule that isn’t disabled.

Create your new schedules like so:

­­­­­­





## Install Python & Required Modules

1. On the Windows host you want to run VizAlerts from, download and install Python 2.7. This can be done in multiple ways, but we suggest this MSI installer: <https://www.python.org/ftp/python/2.7.9/python-2.7.9.msi>

1. Add ";C:\Python27\;C:\Python27\Scripts\" to your Path environment variable (assuming you chose the installation defaults when installing Python)
2. Install the following Python modules:
   1. [PyYAML](http://pyyaml.org/) (recommended: <http://pyyaml.org/download/pyyaml/PyYAML-3.11.win32-py2.7.exe> )
   2. [psycopg2](http://www.stickpeople.com/projects/python/win-psycopg/) (recommended Windows port: <http://www.stickpeople.com/projects/python/win-psycopg/2.6.0/psycopg2-2.6.0.win32-py2.7-pg9.4.1-release.exe> )
   3. The final five packages are best installed by opening a *new* command prompt and running the following commands:  
        
      *pip install requests  
      pip install requests\_ntlm  
      pip install pypdf2  
      pip install phonenumberslite*

*pip install twilio*  
If your computer does not have access to the Internet, see [Appendix A](#_Appendix_A).

## Configure VizAlerts

Now, we can configure VizAlerts. Unzip the VizAlerts.zip file to a folder of your choosing. For the purposes of this manual, we’ll assume the files were extracted to C:\VizAlerts.

### The vizalerts.yaml Configuration File

The next task is to give VizAlerts all the information it needs to connect to our Tableau Server instance. Open the file C:\VizAlerts\config\vizalerts.yaml in a text editor. Each of the configuration settings in that file are commented to explain what they do, so we’ll just go over the most important ones here:

|  |  |
| --- | --- |
| **Email Settings** |  |
| smtp.serv | This is the name of your SMTP server, which VizAlerts will use to send email with. |
| smtp.address.from | The **default** email address you wish email alerts to be sent from. Emails about failures and all Simple Alerts will use this address as the From address. However, users may still customize their own From addresses in Advanced Alerts, which we’ll cover later**.** |
| smtp.address.to | When an alert fails to run, failure details will be sent to this address along with the Subscriber, so it makes the most sense to use your own address or Admin distribution list here. |
| smtp.ssl | When true, VizAlerts will attempt to use SSL for email encryption (which your SMTP server must support). If you do not wish to use encryption, leave it “false”. |
| smtp.user | Username for the account used to connect to your SMTP server. If no authentication is needed, leave it “null” |
| smtp.password | Password for the account used to connect to your SMTP server. If no authentication is need, leave it “null”. The password must be enclosed in single quotes.  If desired, this value can be a valid path to a .txt file containing the password, e.g. 'c:\users\mcoles\password.txt', rather than the password itself. |
|  |  |

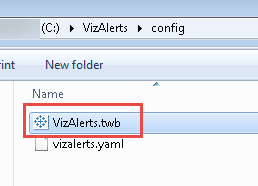
|  |  |
| --- | --- |
| **Tableau Server Settings** |  |
| server | Name of the Tableau Server you wish to run this instance of VizAlerts against. |
| server.version | Major version of the Tableau Server you are running VizAlerts against (this must be 8, 9, or 10) |
| server.user | This is ANY user licensed in Tableau Server--it does not need to be an Admin, as it is only used in authenticating over HTTP.   * If you are using Active Directory authentication, prepend the domain name in front of the username, e.g. “tableau.com\mcoles” * If you are using Local Authentication, simply supply the username, e.g., “mcoles” |
| server.user.domain | This is the domain for the server.user account. Leave this as null (no quotes) if Tableau Server uses local authentication, rather than Active Directory authentication. |
| server.ssl | When set to true, use SSL to connect to Tableau Server (recommended if you have enabled SSL). |
| vizalerts.source.viz | This value is very important. It is the path to the Tableau Server view that you will use to configure all of the rest of the administrative settings for VizAlerts. Must be of the form 'WorkbookURL/ViewUrl', as you’d use to access your viz on Server (as in https://myserver/views/WorkbookURL/viewURL). |
| vizalerts.source.site | Site that the vizalerts.source.viz exists within. Leave as 'Default' if using the default site. |

|  |  |
| --- | --- |
| **SMS Settings** |  |
| smsaction.enable | ‘true’ if you wish to use VizAlerts to send SMS messages; false if you don’t |
| smsaction.provider | The provider you wish to use for SMS messages (must be ‘twilio’ at the moment, it’s the only provider we support) |
| smsaction.account\_id | Account SID from Twilio (see <http://twilio.com/user/account>) |
| smsaction.auth\_token | Your secret authentication token for Twilio (see <http://twilio.com/user/account>). May also be valid path to a plaintext file containing the auth token, e.g. 'c:\users\mcoles\authtoken.txt' |

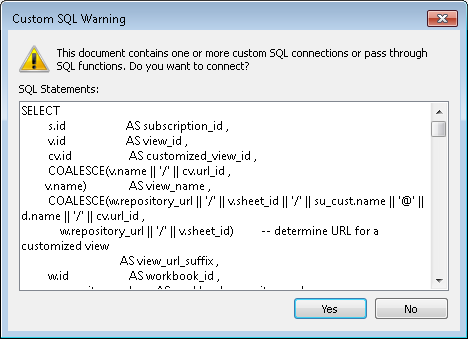
|  |  |
| --- | --- |
| **Processing Settings** |  |
| threads | This is the number of threads VizAlerts will use to process alerts. A higher number will increase concurrent alert processing, ensuring that alerts are run and sent in a timely fashion—but the more alerts run at the same time, the higher a load Tableau Server will be under. A *rough* guideline might be 2 threads for every 50 *total* alerts you have on Server altogether. |

### The VizAlerts Configuration Workbook

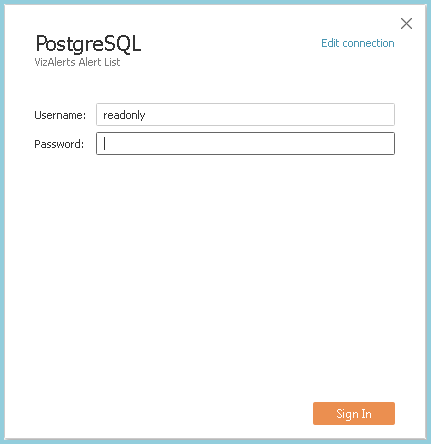
The vast majority of the configuration options for VizAlerts are defined in the configuration workbook. Why a Tableau workbook, rather than in the yaml file we just edited? Because as a workbook hosted on Tableau Server, we have an *amazing* amount of flexibility in defining what alerts are allowed to do what. Let’s open the workbook, located in config\VizAlerts.twb, in Tableau Desktop:



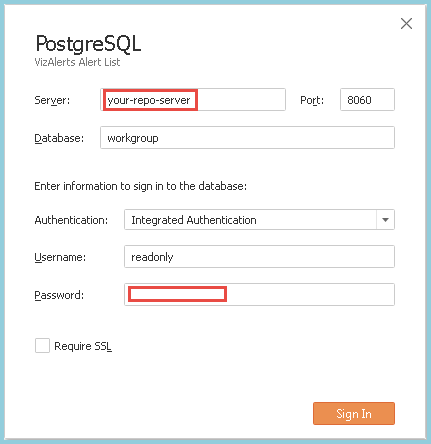
The first thing you’ll get is this:



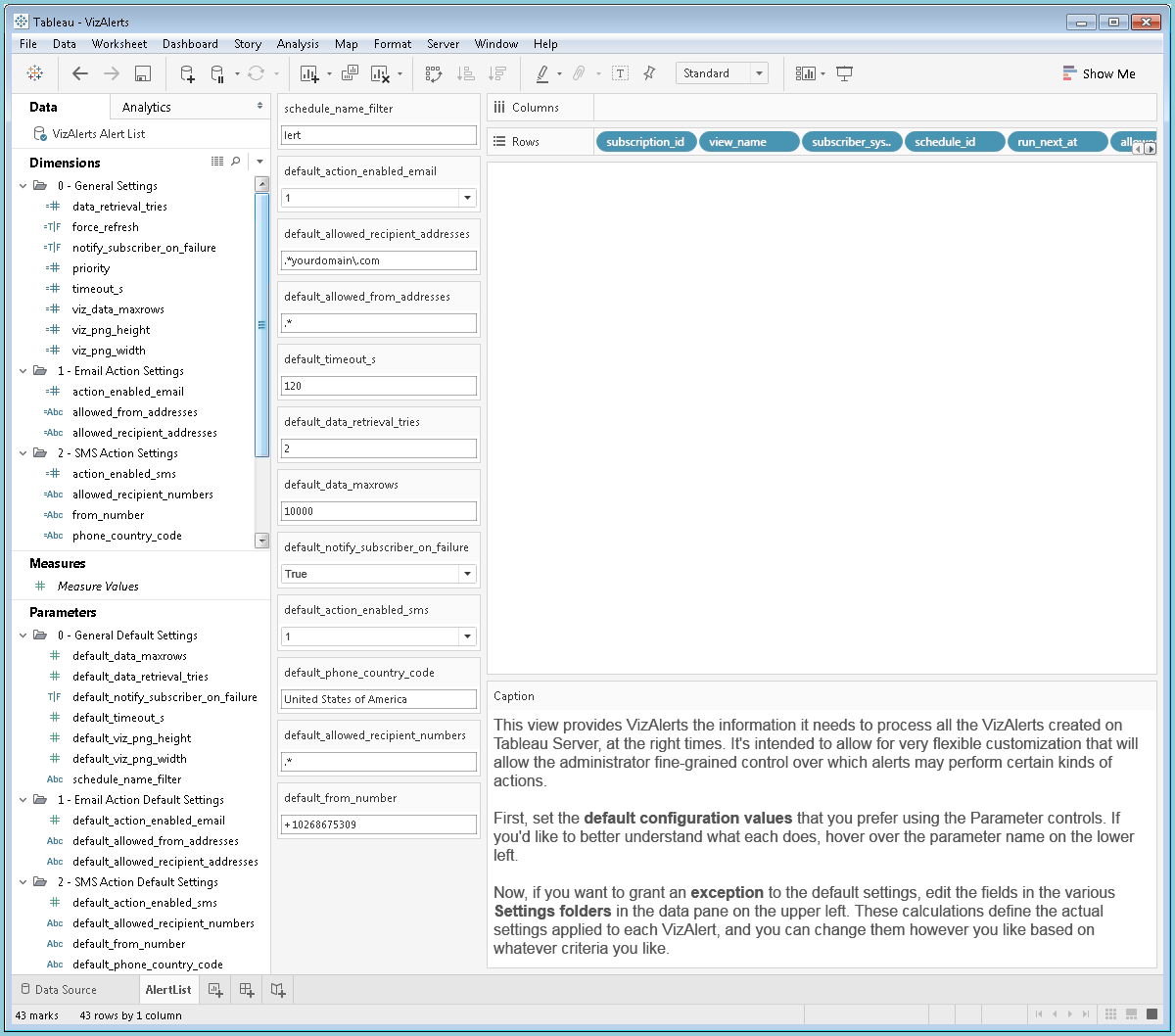
That’s no problem, just click “Yes”. You’ll now see:



Click “Edit Connection” and change the server name to the host that your Tableau Server repository process is hosted on, and change the password to the password you provided in the tabadmin dbpass command [you entered earlier](#_Repository_Access).



You should now see something like this:



The data from this workbook is what VizAlerts looks at to decide *which* VizAlerts on Tableau Server it will process, and *what* options it will use when it does. It has been set up to make configuration easy, and it’s pre-loaded with recommended defaults. But, it’s still strongly recommended to read through each to determine how it works.

**Parameters**

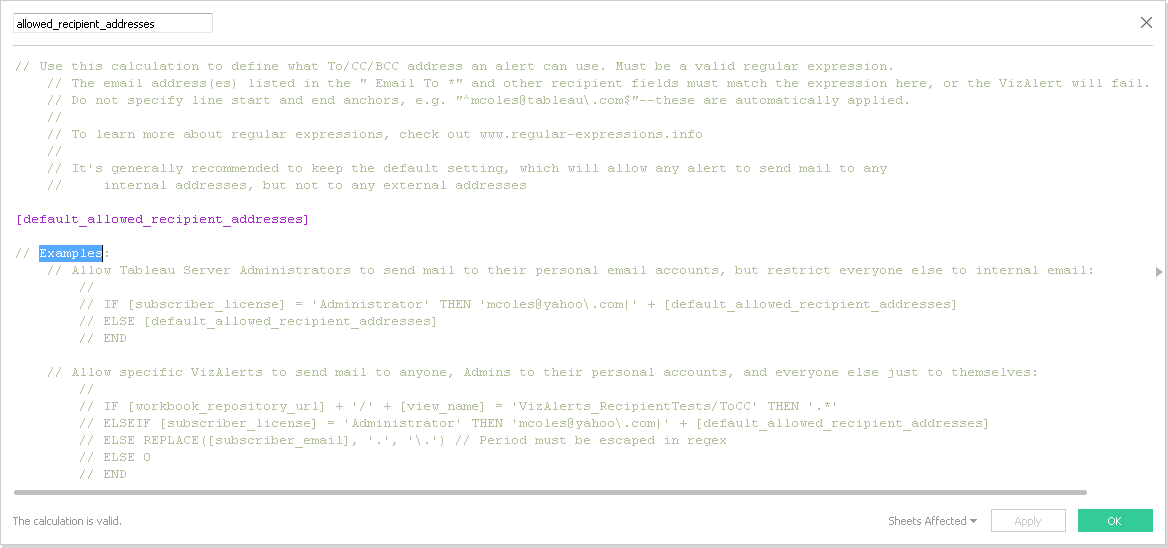
Each of the Parameters is used to define a **default setting** for all VizAlerts. To see what each does, hover over the name in the Parameters pane. We won’t go over all of them in this document, but here are the most important ones you’ll need to know about before getting started:

|  |  |
| --- | --- |
| **schedule\_name\_filter** | This simply tells VizAlerts what your schedule names must contain to be considered true VizAlerts schedules (they must also be disabled) |
| **default\_action\_enabled\_email** | Enables bulk Email Actions for the VizAlert in question (simple alerts are not prevented by this setting) |
| **default\_allowed\_recipient\_address** | Describes what email address(es) a given VizAlert is able to send email to/from/cc/bcc, defined as a regular expression. **Please replace “yourdomain.com” with your company’s domain as a first step**—that will limit VizAlerts to sending email only internally (example: “.\*tableau\.com” would restrict outgoing mail to just tableau.com addresses. |
| **default\_allowed\_from\_address** | Authors of Advanced Alerts can specify what address they want their emails to show as having been sent from. This works the same way as the allowed\_recipient\_address parameter does. **Please replace “yourdomain.com” with your company’s domain as a first step**—that will limit VizAlerts mail to sending email only from vizalerts@yourdomain.com. |
| **default\_action\_enabled\_sms** | Enables SMS Actions for the VizAlert in question (you must have set up vizalerts.yaml for SMS messaging to enable them, first) |
| **default\_phone\_country\_code** | If authors of an SMS message don’t provide destination numbers with country codes (in E.164 format), this is the code that will be used. |
| **default\_allowed\_recipient\_numbers** | Describes what phone numbers a given VizAlert is able to send SMS messages **to**, defined as a regular expression.  The E.164 version of whatever number(s) are specified in the "SMS To \*" field in an alert are what will be evaluated against this regex, *not* the raw text the alert author used (if they entered “(206) 867-5309”, the regex will be evaluated against “+12068675309”) |
| **default\_from\_number** | Describes what Twilio phone number or Message Service SID a given VizAlert will use to send SMS messages **from**.  If a phone number is used, it must be in E1.64 format, e.g. "+12068675309".  If a Message Service SID is used, it must be the full string, e.g. "MG9752274e9e519418a7406176694466fa" |

**Calculations**

What if one of your alert authors wishes to send emails to audiences outside your organization? Or use a custom “from” email alias? Or what if you want to make sure all SMS messages from Marketing use a special higher-volume Message Service in Twilio? You can make **exceptions** to the default values you set in the Parameters, based on any information in the configuration workbook, simply by editing the appropriate *calculations*. Because the information in a calculation is dynamic, you can use the other fields such as project\_name, owner\_sysname, schedule\_frequency, etc to change how a given alert is processed. And if that flexibility wasn’t enough, you can get even more advanced and blend in data from other sources, or use the cross-database join feature in order to customize your field data to an absurd degree.

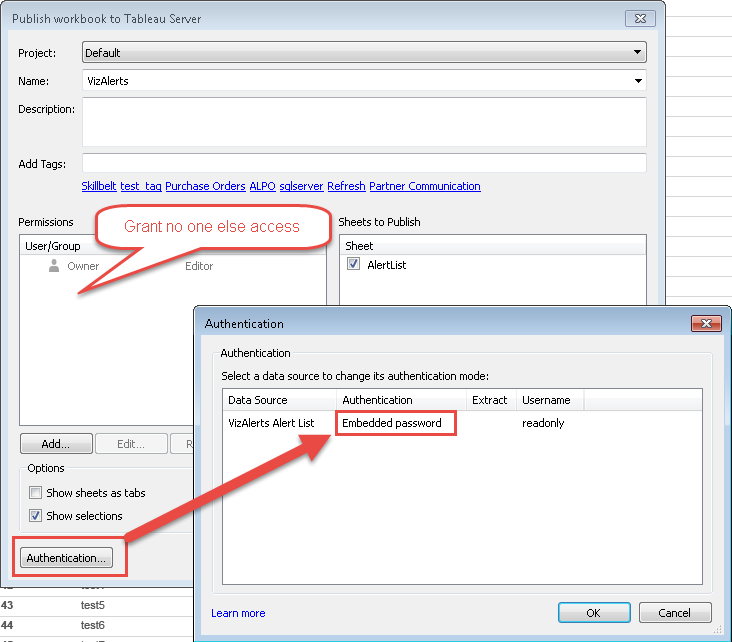
Again, we won’t go over all the calculations in this document, as they’re well-documented in the workbook itself. But, here is an example:



**Publish!**

Once you feel like you have the settings you want, you need to publish the workbook to Tableau Server. When you publish it, make sure you do two things:

* **Ensure that only Administrators have edit rights!**
  + You don’t want unauthorized individuals granting themselves exceptions to the settings you created.
* **Embed credentials**
  + VizAlerts will fail if you don’t embed them properly



## Testing

Whew! All that was lots of fun, but let’s get to the good stuff and test this thing to see if we did everything right. We’ve got a few tests to run to validate that everything is working, starting out from simple to more complicated:

### Can VizAlerts Connect? Test

Run the following from a command prompt on the Windows host you set VizAlerts up on. By default, VizAlerts will expect you are running it within the context of the directory you created it in, so change to that directory first, then run the script:

cd C:\VizAlerts

python C:\VizAlerts\vizalerts.py

It should have successfully generated a Trusted Ticket, queried the PostgreSQL database in Tableau Server, then realized there was nothing to do and quit without error. If it didn’t, please see the [Common Errors](#_Common_Errors:) section.

### Simple Alert Test

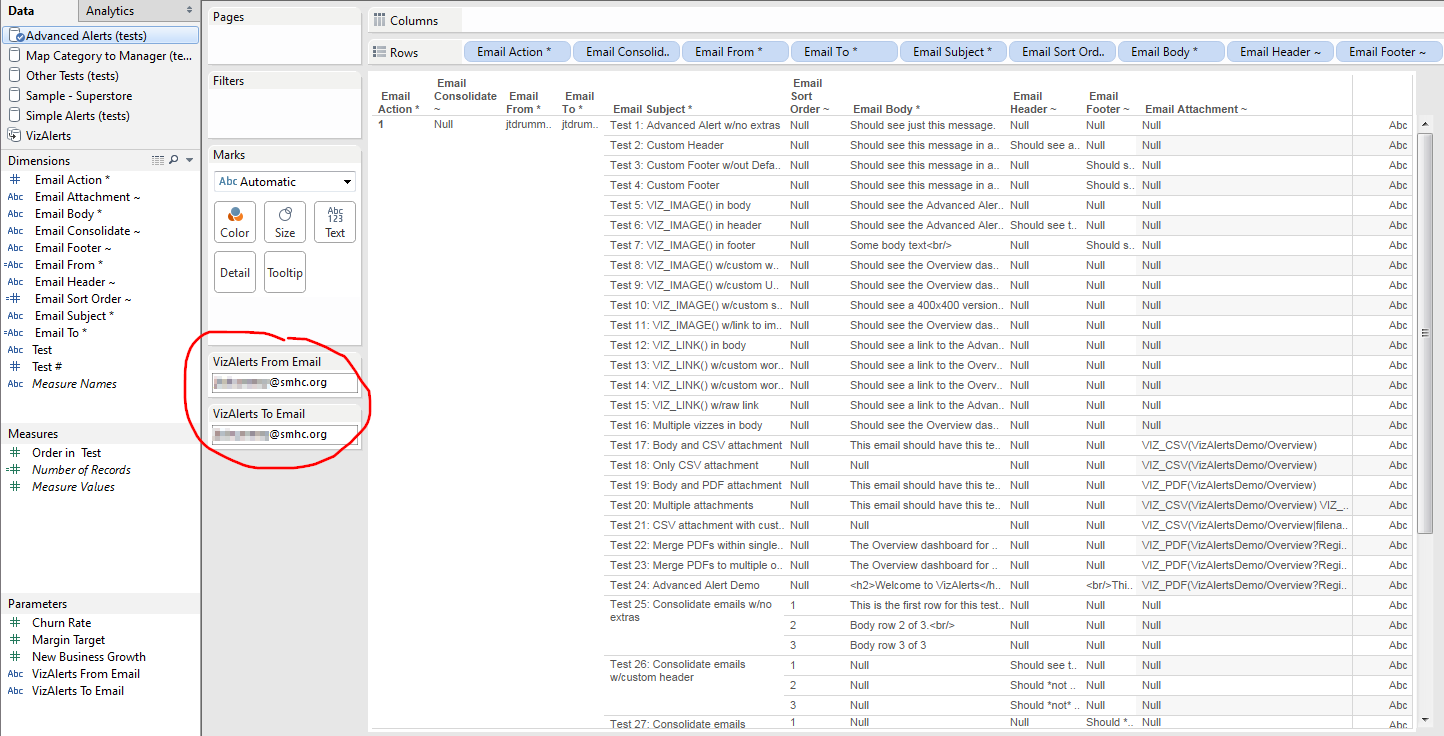
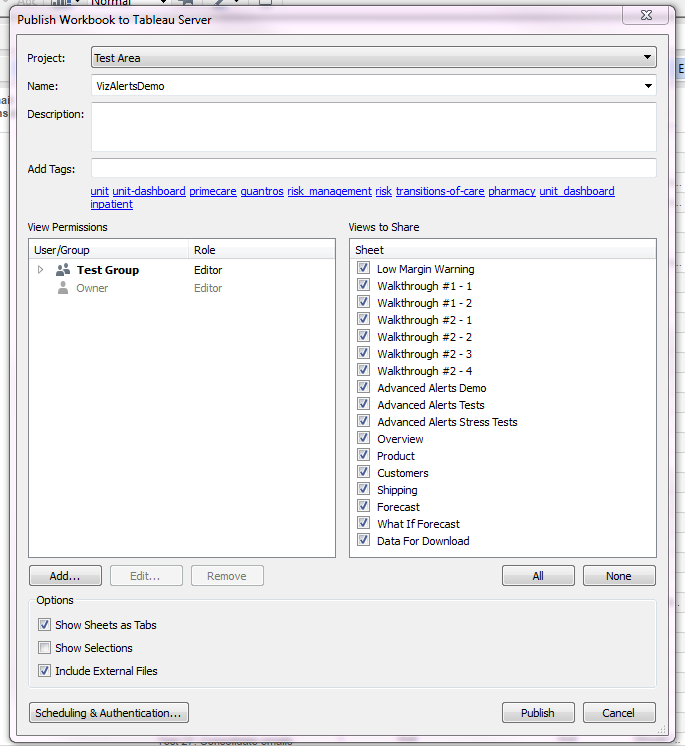
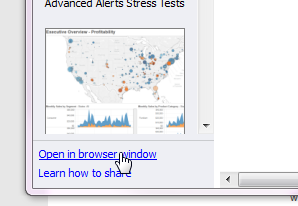
Now for a more extensive test on a Simple Alert. Subscribe to any Tableau Server View on a VizAlerts schedule that you set up (pick a view that renders in less than 10 seconds or so). I recommend subscribing on a VizAlerts schedule that runs every 15 minutes for this test, even if you just remove it afterward. After you subscribe, run the command again:

python C:\VizAlerts\vizalerts.py

Now, wait 15 minutes, then run the same command again. If data is present in the viz, you should receive an email! If not, you shouldn’t. Simple as that!

### Put VizAlerts Through Its Paces Test

For this test you are going to use the same Tableau workbook that the VizAlerts contributors use to verify VizAlerts is working after we’ve changed the code. Note that this workbook only works with Tableau version 9.0 and up.

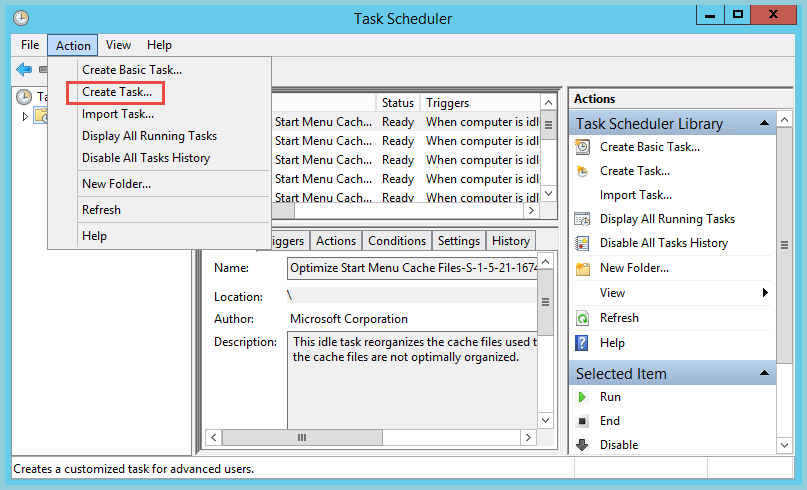
1. In Tableau Desktop open [VizAlerts Install Folder]\VizAlerts\demo\VizAlertsDemo.twb.
2. Go to the **Advanced Alerts Tests** view and set the VizAlerts From Email and VizAlerts To Email parameters to your test email address:  
     
   
3. Choose Server->Publish workbook… to start the publishing process. Use the default settings, which will include the External Files option:  
     
     
     
   We suggest you publish the workbook in a place where other users who will be configuring Advanced Alerts (see the User Guide) can see the workbook.
4. Click through the warning(s) about including external files and publish the workbook.
5. When the confirmation window appears, click Open in browser window to open the VizAlertsDemo workbook on Tableau Server.  
   
6. Login to Tableau Server if you need to and navigate to the Advanced Alerts Demo worksheet.
7. Scroll down in the worksheet and enter a comment with the text “test\_alert”.
8. After the comment has been posted, go back to your Windows command prompt and enter:  
     
   python C:\VizAlerts\vizalerts.py  
     
   If the script runs and exits the first time without processing anything, run it again. (Tableau can take a moment to update the data with the “test\_alert” comment that acts as a trigger). VizAlerts will now generate **30+** emails with a variety of tests demonstrating the VizAlerts features. Read through the emails to understand what is expected of each. If you get any error messages then check the Common Errors section below as well as the FAQ in the User Guide.

## Final Steps

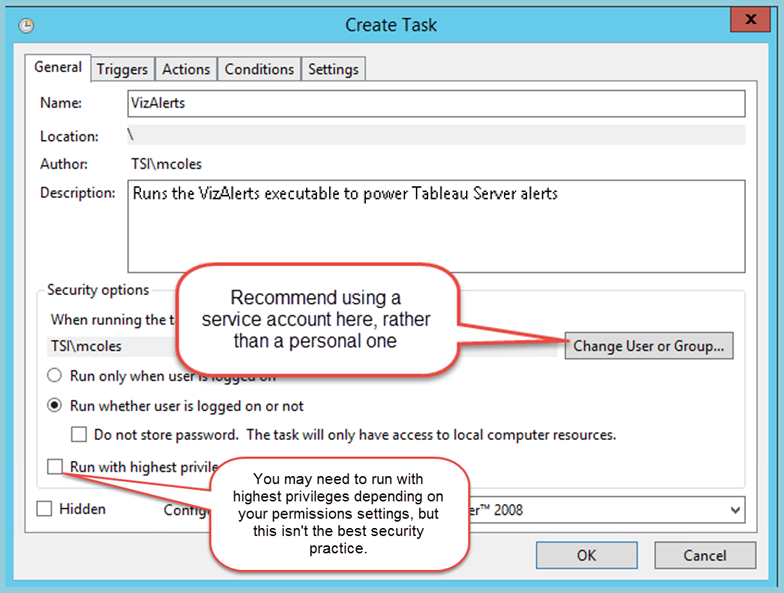
### Set up a Scheduled Task

The last step, now that everything is working as expected, is to automate this so that VizAlerts can run regularly when it is supposed to. To do this, we need to set up a Scheduled Task on the Windows host that VizAlerts runs from, which will run this for us on a regular basis.

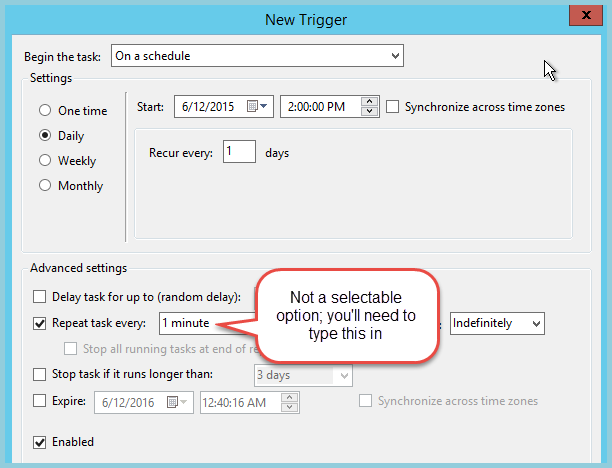
First, let’s create a new Task:



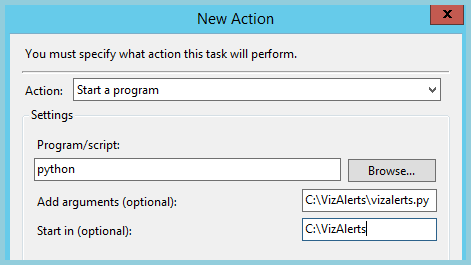
Fill in the name and description. Make sure it will run whether the user is logged in or not. The task should be set up to run under a service account rather than a personal one, if possible. This account must have full control permissions on the VizAlerts files, and if you specified text files instead of passwords in the vizalerts.yaml config file, the account will need rights to read those files.



Set up the Trigger (when will it run?). We strongly recommend running this every **1 minute**, as this will keep alerts executing on time, and the vast majority of executions will be quick checks that don’t actually do any work. Additionally, it will ensure that anyone testing their alerts will get a quick result.



Set the Action on the Task (what will it do?)



### Helper Datasource

Last, but not least, publish the …\demo\VizAlerts.tdsx data source to Tableau Tableau Server, and grant permissions to anyone you wish to have an easier way to create Advanced Alerts. This datasource simply gives users a shortcut to creating them (see the User Guide for more details).

### 

# FAQ

* **How many alerts can be run at once?**Alerts are processed in parallel, according to the number of threads you set in the config\vizalerts.yaml file. They are checked according to the Schedule they are associated with, in order of the “priority” field in the config workbook. Alerts scheduled for 6AM will begin being checked at 6AM, so if several long-running alerts are checked at that time, subsequent alerts may be checked sometime after 6AM. This can be exacerbated if long-running alerts aren’t removed from the pipeline, and/or reasonable timeout settings set in the config workbook.
* **Does VizAlerts use a database to log information about what it has done?**No, not in its current state, though this is the next logical progression for it. Currently it logs information into text files only. For SMS, the Twilio does log all messages that are sent, and this information can be downloaded and analyzed.

# Common Errors

* **Failed with unknown protocol**
  + This likely means that you’ve enabled SSL in the vizalerts.yml, but haven’t set Tableau Server up for it. See [this portion](http://onlinehelp.tableau.com/current/server/en-us/ssl_config.htm) of the online help on how to do so.
* **Parse errors**
  + Generally this means that some bad character or formatting issue was introduced to the vizalerts.yaml file (typically a tab character—replace them with spaces!). I recommend using this [online YAML validator](http://codebeautify.org/yaml-validator) to determine where the problem is (make sure to remove your passwords first!).
* **Postgres connection failure**
  + Ensure the [readonly user](http://onlinehelp.tableau.com/current/server/en-us/adminview_postgres_access.htm) is set up for PostgreSQL repository access, and that the password you’ve specified is correct.
* **Trusted ticket failure**
  + Check to ensure your trusted tickets were [configured properly](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth_trustIP.htm). If things are still not working, try [this article](http://kb.tableau.com/articles/knowledgebase/testing-trusted-authentication) to test them further.
  + A “-1” result could be due to several possible issues. Please see [this article](http://onlinehelp.tableau.com/current/server/en-us/trusted_auth_trouble_1return.htm) if you’re seeing this error.
* **Unable to export … as CSV**
  + This means that the attempt to export the view data for an alert to a CSV file failed, either because of internal errors, or because it took longer than the timeout you’ve set in the config workbook. If the view can be rendered successfully in your browser, it may simply be taking too long. Increasing the timeout settings may help with this, but a better solution is to try and optimize the viz to render more quickly.
  + If you see a **406 HTTP error, aka “Not Acceptable”** it means that Tableau Server itself could not load the viz. That could be any one of the following reasons:
    - The subscriber does not have access to the view
    - The view could not connect to its datasource for some reason
    - A Tableau Server process crashed when it tried to load the view
    - The view had an invalid calculation and couldn’t be loaded

# 

# Getting VizAlerts Help

First of all, check with any local admins and any local documentation that might exist. After that, the center for all things VizAlerts is the VizAlerts Group on the Tableau Community <https://community.tableau.com/vizalerts>.

# Contributing to VizAlerts

VizAlerts is an open source project distributed under the MIT License. If you’d like to contribute ideas or code to VizAlerts, please visit the VizAlerts GitHub site at <https://github.com/tableau/VizAlerts>.

# Appendix A

## Installing Python modules with no Internet access

Setting VizAlerts up on a secure machine that isn’t connected to the Internet can be done by following these instructions. It essentially requires that you download the files that you need from a machine that is connected to the Internet, then copy them over to the secured machine you’ll be running VizAlerts from.

1. First, download the Python [install file](#_Install_Python_&). Install it on the Internet-connected machine you’re using to download files, then copy it to your VizAlerts host and install Python there too. On both machines, you may wish to follow Step 2 as well, and add the Python executables to your PATH environment variable.
2. From your Internet-connected machine, run the following commands to download all of the required Python modules (feel free to adjust the path they download to). These function as basically offline package repositories:

*pip install --download c:\mypythonpackages requests*

*pip install --download c:\mypythonpackages requests\_ntlm*

*pip install --download c:\mypythonpackages pypdf2*

*pip install --download c:\mypythonpackages phonenumberslite*

*pip install --download c:\mypythonpackages twilio*

1. Copy the entire folder to your offline machine (I'm assuming here that it's copied to the same path).
2. On your offline machine, install the package from the newly copied folder:

*pip install --no-index --find-links file:c:\mypythonpackages requests*

*pip install --no-index --find-links file:c:\mypythonpackages requests\_ntlm*

*pip install --no-index --find-links file:c:\mypythonpackages pypdf2*

*pip install --no-index --find-links file:c:\mypythonpackages phonenumberslite*

*pip install --no-index --find-links file:c:\mypythonpackages twilio*

1. Check for errors in the output. If there are none, you’ve successfully got Python and all the modules installed!