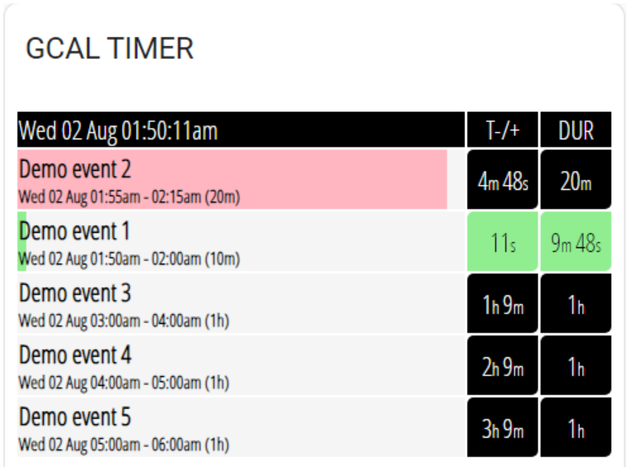
Guide for gcal_timer (Home Assistant / AppDaemon)
Ravichandran Subramanian / ravisdxb@gmail.com

Purpose:

To extract Google Calendar events and show them with a timer and a progress bar. Each user could see only their own calendar, if set up correctly.



GCAL TIMER

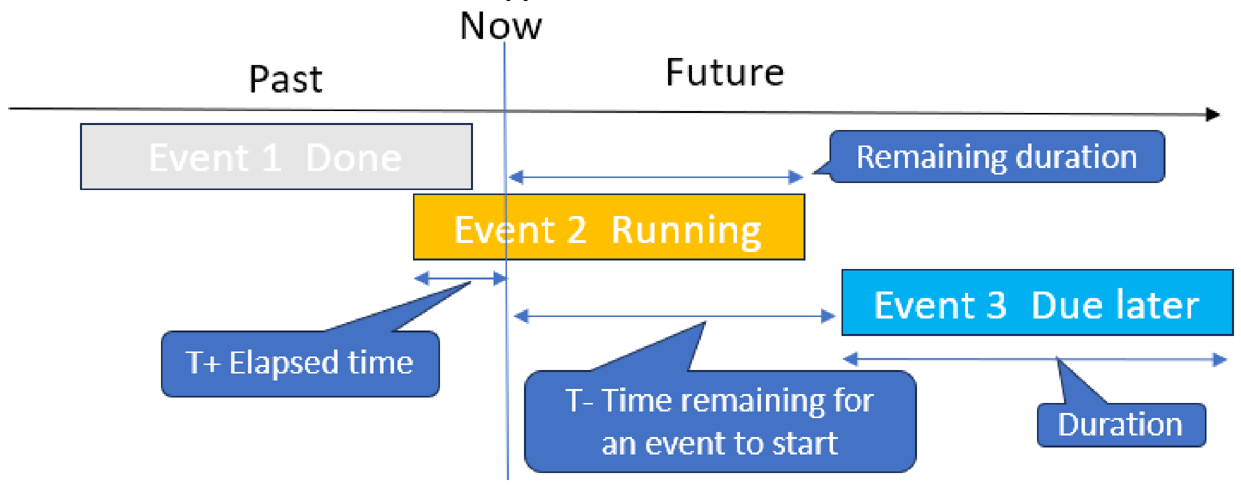
Wed 02 Aug 01:50:11am	T-/+	DUR
Demo event 2 Wed 02 Aug 01:55am - 02:15am (20m)	4m 48s	20m
Demo event 1 Wed 02 Aug 01:50am - 02:00am (10m)	11s	9m 48s
Demo event 3 Wed 02 Aug 03:00am - 04:00am (1h)	1h 9m	1h
Demo event 4 Wed 02 Aug 04:00am - 05:00am (1h)	2h 9m	1h
Demo event 5 Wed 02 Aug 05:00am - 06:00am (1h)	3h 9m	1h

Event starts in less than 5m.

Running: shows elapsed and remaining time

Events 3,4 & 5 are due later.

Some of the conventions used in this app:



Important considerations:

1. Need AppDaemon, HACS, and state-switch custom component.
2. This app uses state-switch custom component to show each user only their own calendar. Since it should be added manually, the admin user should take control of dashboard, after which, automatic addition of entities to the dashboard will not work. Please do not proceed with installation of this app if you are not willing to take control of dashboard.
3. Since this app accesses Google Calendar from Home Assistant/AppDaemon, each user should obtain Google credentials and complete the OAuth flow, which involves consenting to Home Assistant accessing their Google Calendar. This is done on a desktop/laptop for the first time. For this Python should be installed on the system. AppDaemon will handle the periodical refresh of the list of events from second time onwards.
4. The app provides some user configurable parameters (explained later).

Setup:

Setting up gcal_timer involves lot of "things" to do, which are broadly grouped into:

A. Things to do on Google platform, which includes obtaining a Google Calendar credential, setting up an OAuth consent screen from the web and completing the Google OAuth flow on a PC to generate a token from the credential. If this was done earlier for some other purpose and `token.json` file is already available, please proceed to B.

B. Things to do on Home Assistant, which includes copying the required files to specific directories and making the gcal_timer appear on the Home Assistant dashboard.

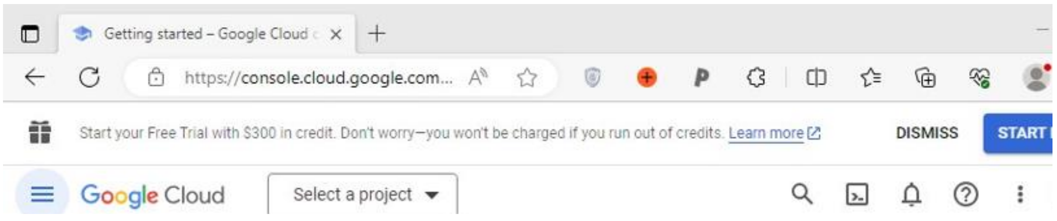
A. Things to do on the Google Platform (for each user)

The guide for this task available at Google Calendar > Python QuickStart (https://developers.google.com/calendar/api/quickstart/python)

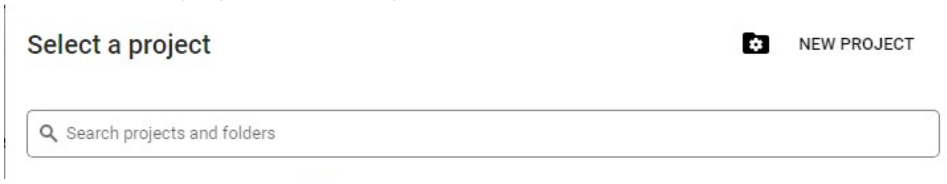
A.1. Setup a project

A.1.1. Log into Gmail. This is not mandatory. However, this makes it easier since it logs the user into any other Google app without having to enter ID and password again.

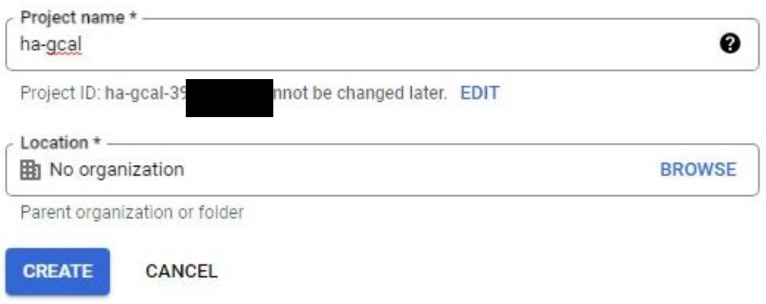
A.1.2. Log into Google Cloud Console (https://console.cloud.google.com)


Getting started – Google Cloud
https://console.cloud.google.com...
Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
DISMISS
START
Google Cloud
Select a project

A.1.3. Select a project > New Project



A.1.4. Enter a project name and click CREATE. Any name is OK. I have named it ha-gcal to mean it is for Home Assistant – Google Calendar. This name is not used anywhere else other than Google Cloud.



Project name *
ha-gcal

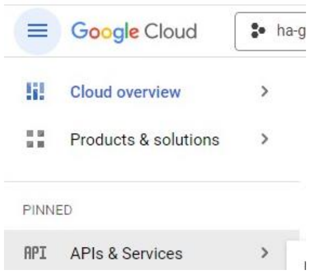
Project ID: ha-gcal-39[redacted]nnot be changed later. EDIT

Location *
No organization BROWSE

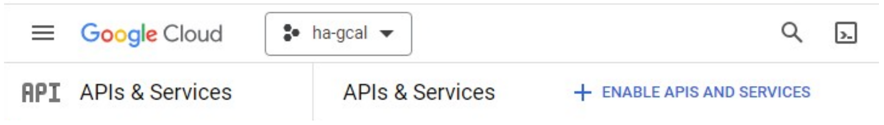
Parent organization or folder

A.2 Enable Google Calendar API

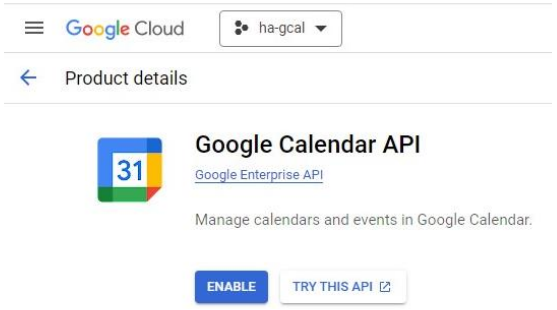
A.2.1. From the triple line menu (hamburger icon), select APIs & Services



A.2.2. From API menu, select Enable APIs and Services

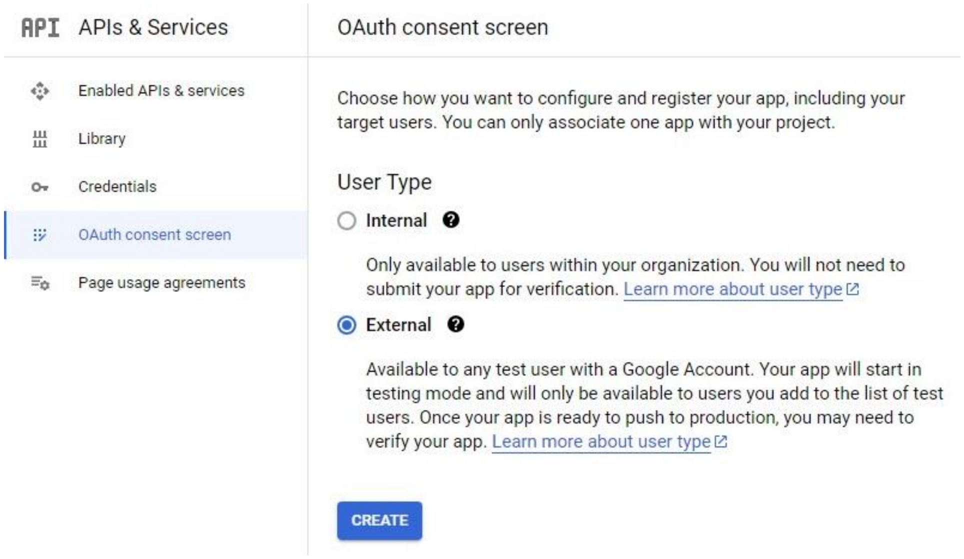


A.2.3. Search for Google Calendar API and enable it.



A.3 Configure OAuth consent screen

A.3.1. From API menu, select OAuth consent screen > User type: External > CREATE



OAuth consent screen

Choose how you want to configure and register your app, including your target users. You can only associate one app with your project.

User Type

Internal

Only available to users within your organization. You will not need to submit your app for verification. Learn more about user type

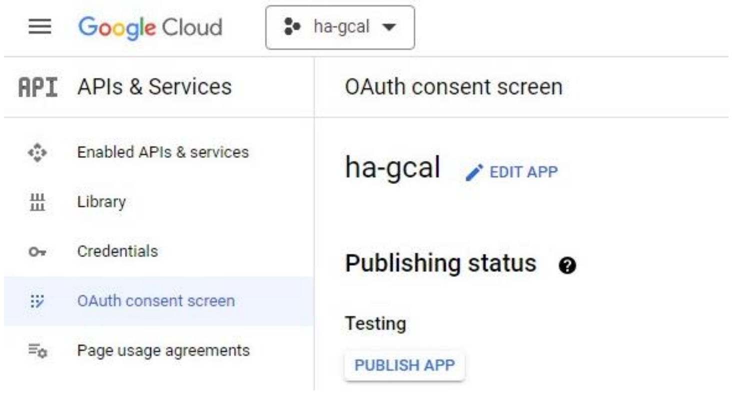
[✓ External]

Available to any test user with a Google Account. Your app will start in testing mode and will only be available to users you add to the list of test users. Once your app is ready to push to production, you may need to verify your app. Learn more about user type

A.3.2 Provide user support email and Developer contact information (these two fields are mandatory and marked with a * - enter your Gmail id in both fields)

A.3.3 Proceed without any changes in Scopes and Optional Info pages.

A.3.4 In the Summary page, click on PUBLISH APP to move from Testing status to Production status.



OAuth consent screen

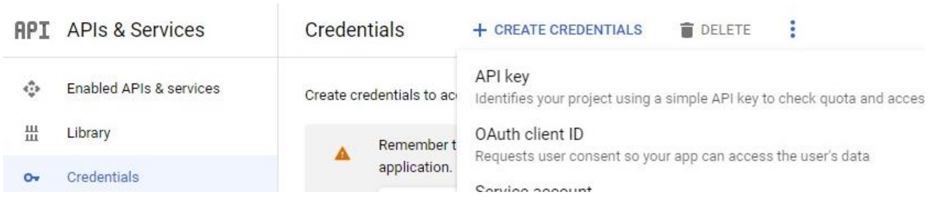
ha-gcal

Publishing status

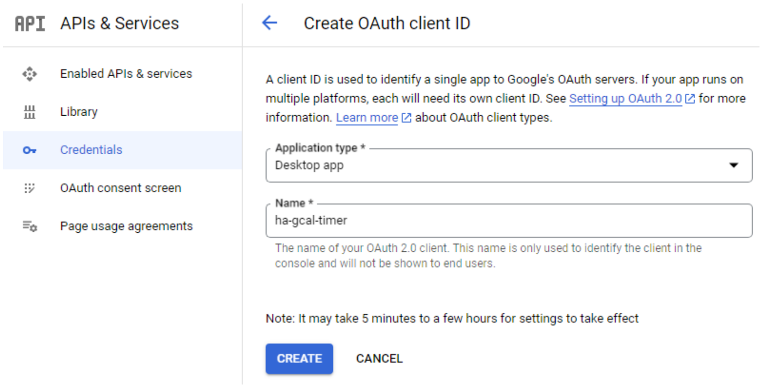
Testing

A.4 Obtain an OAuth credential

A.4.1. From the API menu, select Credentials > Create Credentials > OAuth client ID



A.4.2. Select Application type Desktop app, give it a name and then click CREATE



A client ID is used to identify a single app to Google's OAuth servers. If your app runs on multiple platforms, each will need its own client ID. See Setting up OAuth 2.0 for more information. Learn more about OAuth client types.

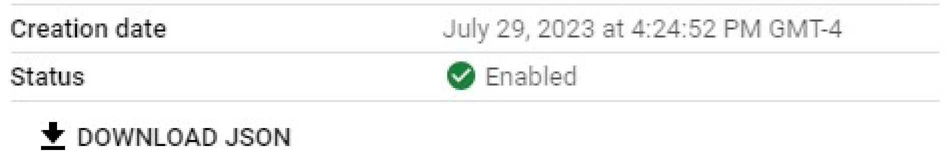
Application type * Desktop app

Name * ha-gcal-timer

The name of your OAuth 2.0 client. This name is only used to identify the client in the console and will not be shown to end users.

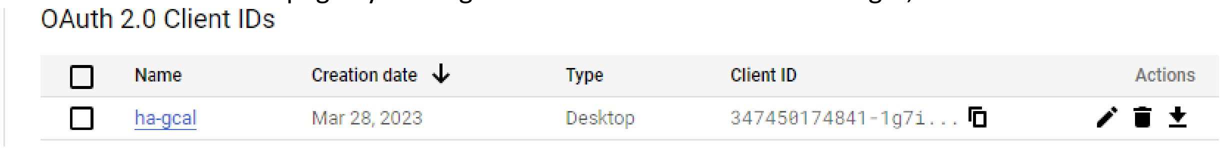
Note: It may take 5 minutes to a few hours for settings to take effect

A.4.3. Download the credentials file either from the next screen immediately after creation, by clicking the DOWNLOAD JSON link



Creation date	July 29, 2023 at 4:24:52 PM GMT-4
Status	Enabled

or from the Credentials page by clicking on the downward arrow at the right, below Actions.


OAuth 2.0 Client IDs

	Name	Creation date	Type	Client ID	Actions
☐	ha-gcal	Mar 28, 2023	Desktop	347450174841-1g7i...	

The JSON file is named `credentials.json` and will be available in your Downloads folder.

A.5 Complete OAuth flow on PC/laptop

A.5.1. Download and install Python, if not already done (https://www.python.org/downloads/)

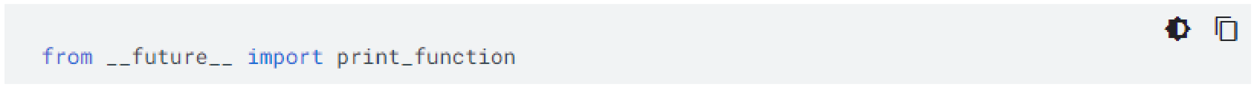
A.5.2. Create a new folder oauth and copy the credentials.json from Downloads to oauth.

A.5.3. Start a command prompt and change directory to oauth

A.5.4. Install the Google client library for Python, by running the following command at c:\oauth>

```
pip install --upgrade google-api-python-client google-auth-httplib2 google-auth-oauthlib
```

A.5.6. From https://developers.google.com/calendar/api/quickstart/python copy the Python code and save it under oauth folder with the name `oauth-gcal.py`.



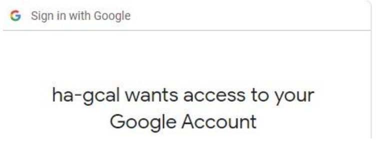
```
from __future__ import print_function
```

A.5.7. Run the Python code at c:\oauth>

```
python oauth-gcal.py or python3 oauth-gcal.py
```

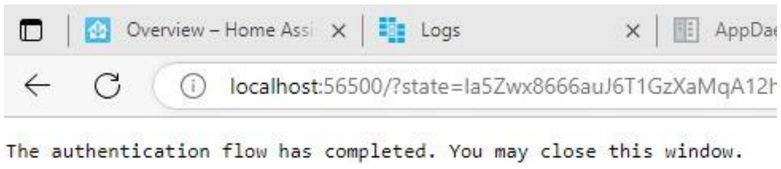
A.5.8. Click on the unsafe link at the bottom of the "Google has not verified this app" page.

A.5.9. Click Continue on the next page, which says your project wants to access your Google Account.



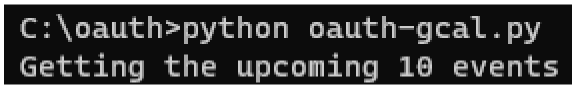
ha-gcal wants access to your Google Account

A.5.10. The authentication flow is completed.



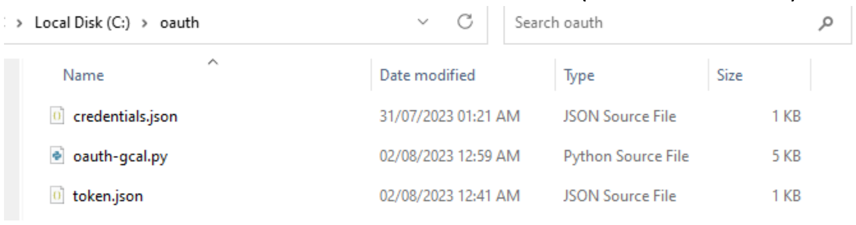
```
The authentication flow has completed. You may close this window.
```

A.5.11. If everything goes well, a list of 10 Google Calendar events are listed in the Command window. Also, a new file token.json is created in the oauth folder. At this stage, the oauth folder will have 3 files.



```
C:\oauth>python oauth-gcal.py
Getting the upcoming 10 events
```

(events truncated)



Name	Date modified	Type	Size
credentials.json	31/07/2023 01:21 AM	JSON Source File	1 KB
oauth-gcal.py	02/08/2023 12:59 AM	Python Source File	5 KB
token.json	02/08/2023 12:41 AM	JSON Source File	1 KB

A.5.12. If the browser does not popup with the consent page, copy+paste the URL which appears at the command window to a browser and then continue with consent.

B. Things to do on Home Assistant (As the admin user)

B.1. Add an input_boolean helper. This is the trigger to refresh Google calendar events either manually or at pre-defined intervals.
Settings > Devices & Services > Helpers > Create Helper > Toggle
Name: gcal_udpate
Icon: mdi:calendar-clock (or similar)

B.2. Install AppDaemon (if not already done)
Settings > Add-ons > ADD-ON STORE > AppDaemon (from Home Assistant Community Add-ons section)

B.3. Ensure time_zone is correctly identified in /config/appdaemon/appdaemon.yaml.
Please refer to https://gist.github.com/heyalexej/8bf688fd67d7199be4a1682b3eec7568 for a list.

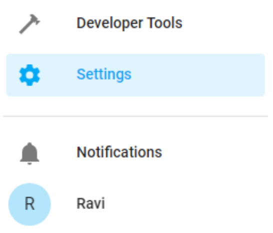
```
---
secrets: /config/secrets.yaml
appdaemon:
  latitude: <lat of your location>
  longitude: <lon of your location>
  elevation: 0
  time_zone: America/New_York
  plugins:
    HASS:
      type: hass
http:
  url: http://127.0.0.1:5050
admin:
api:
hadashboard:
```

B.4. Add the following app specification to the end of config/appdaemon/apps.yaml.

```
gcal_timer:
  module: gcal_timer
  class: GCalTimer
  EVENTS_LIMIT: 15
  REFRESH_MINUTES: 60
  TRIGGER: input_boolean.gcal_update
  USERS:
    - Ravi
    - James Bond
```

Purpose of the parameters:

```
EVENTS_LIMIT: 15 <<number of events retrieved, irrespective of date range. Change as required>>
REFRESH_MINUTES: 60 <<refresh interval in minutes. Change to smaller number for more frequent refresh>>
TRIGGER: input_boolean.gcal_update <<toggle from 5.1 - toggled automatically at refresh interval or manually>>
USERS: <<list of users. Name displayed at bottom left of Dashboard. Required even if single user>>
  - Ravi <<user name is single word>>
  - James Bond <<user name with more than one word. Maintain the space between words>>
```



B.5. Copy the required files to various directories.

B.5.1. Download the repository from GitHub and extract the contents to a folder.
https://github.com/ravisdxb/gcal_timer > Code > Download ZIP
This creates a folder gcal_timer-main in the extraction folder.

B.5.2. Copy the files to relevant AppDaemon folders. Folders which are not present should be created. Please ensure that the copying does not overwrite any of your existing files.

From: `gcal_timer-main\config\appdaemon\apps\gcal_timer.py`
To: `/config/appdaemon/apps/gcal_timer.py`

From: `gcal_timer-main\config\appdaemon\lib` (entire contents of folder)
To: `/config/appdaemon/lib`

From: `gcal_timer-main\config\www` (entire contents of folder – fonts and lib folders)
To: `/config/www`

B.5.3. Create a new directory `/config/appdaemon/gcal` and copy credentials.json and token.json files generated at step# A.5.11.

B.5.4. Rename the `credentials.json` and `token.json` files with the prefix of user name, as below.
For user Ravi, it should be `Ravi_credentials.json` and `Ravi_token.json`
For user James Bond, it should be `James_Bond_credentials.json` and `James_Bond_token.json`

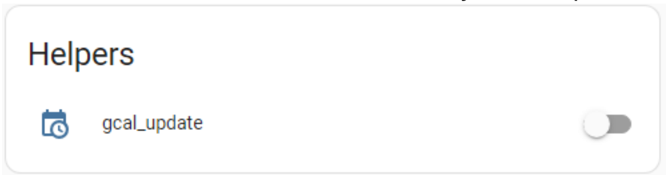
B.5.5. Create a new directory `/config/www/gcal_timer`
The HTML files which show up on the Dashboard will be generated in this location.

B.6. Add gcal_timer to the HA Dashboard.
This involves taking control of the HA Dashboard, after which newer entities will not be added automatically to the Dashboard. Please proceed at your own discretion.

B.6.1. Download and install HACS. https://hacs.xyz/
Please proceed only if you know what you are doing, because it comes with the following statements:
*Everything you find in HACS is **not** tested by Home Assistant, that includes HACS itself.*
*The HACS and Home Assistant teams do not support **anything** you find here.*

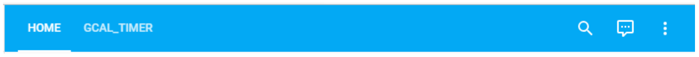
B.6.2. Install state-switch custom component.
This card allows switching the display of certain elements based on the currently logged in user. This enables each user to see only their own calendar (privacy).
HACS > Frontend > + EXPLORE & DOWNLOAD REPOSITORIES
Search for state-switch and install

B.6.3. Toggle `input_boolean.gcal_update` to ON from the Dashboard. The toggle reverts back to OFF if the execution is successful. In case the toggle is stuck in ON position for more than a few seconds, set it OFF manually and repeat the toggling AFTER restarting AppDaemon.

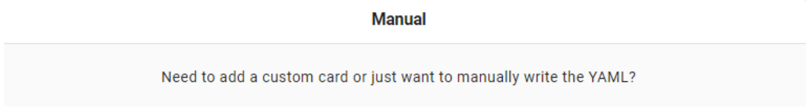


This generates the following 2 HTML files for each user in the folder /config/www/gcal_timer
For user Ravi, `Ravi_gcal_timer_big.html` and `Ravi_gcal_timer_small.html`
For user James Bond, `James_Bond_gcal_timer_big.html` and `James_Bond_gcal_timer_small.html`
_small.html is meant for adding as a card, suitable for mobile devices.
_big.html is meant for adding as a panel, suitable for tablets and TV screens
The contents are essentially the same, except for the font size.

B.6.4. Configure state-switch card and add it to the Dashboard.
Click on the 3 vertical dots at top right and select Edit Dashboard.



Click on + ADD CARD link at bottom right and select Manul at the bottom of available cards.

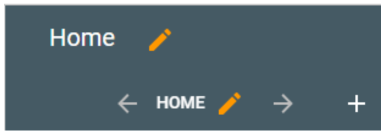

Manual
Need to add a custom card or just want to manually write the YAML?

and copy+paste the following contents into the card configuration and Save.

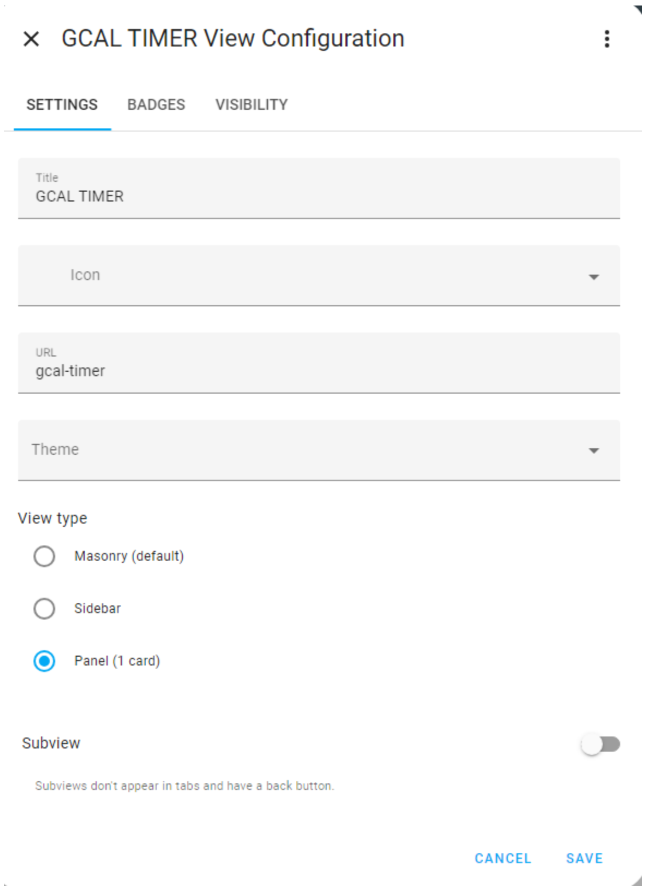
```
type: custom:state-switch
entity: user
default: default
states:
  Ravi:
    type: iframe
    url: /local/gcal_timer/Ravi_gcal_timer_small.html
    title: GCAL TIMER
    aspect_ratio: 160%
  Mangala:
    type: iframe
    url: /local/gcal_timer/James_Bond_gcal_timer_small.html
    title: GCAL TIMER
    aspect_ratio: 160%
  default:
    type: markdown
    content: |
      ## GCAL Timer Not Configured
```

Now, the Dashboard should be showing the list of Google Calendar events with a timer.

To add the gcal_timer as a panel (full page) suitable for larger devices, click on + icon in the header.



and configure it as Panel (1-card) as below:


GCAL TIMER View Configuration

SETTINGS BADGES VISIBILITY

Title: GCAL TIMER

Icon

URL: gcal-timer

Theme

View type

- Masonry (default)
- Sidebar
- [✓ Panel (1 card)]

Subview

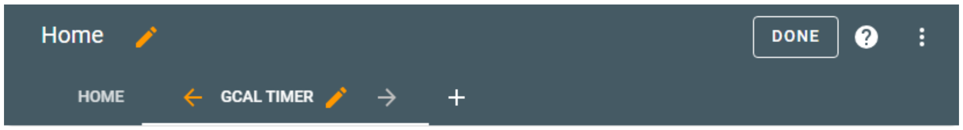
Subviews don't appear in tabs and have a back button.

CANCEL SAVE

Click on + ADD CARD at right bottom, select Manual from the list of available cards and then copy+paste the following contents into the card configuration and Save. The URL now has _big instead of _small.

```
type: custom:state-switch
entity: user
default: default
states:
  Ravi:
    type: iframe
    url: /local/gcal_timer/Ravi_gcal_timer_big.html
    title: GCAL TIMER
    aspect_ratio: 100%
  Mangala:
    type: iframe
    url: /local/gcal_timer/James_Bond_gcal_timer_big.html
    title: GCAL TIMER
    aspect_ratio: 100%
  default:
    type: markdown
    content: |
      ## GCAL Timer Not Configured
```

Click on DONE to save the configuration and display the Google Calendar events.



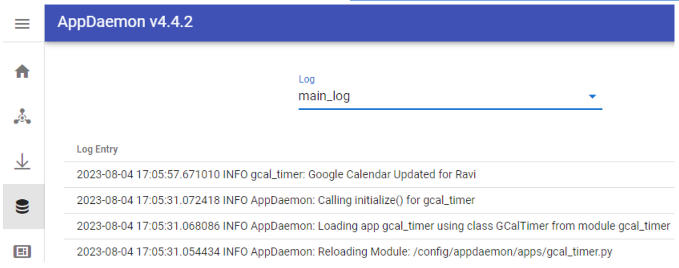
C. Conclusion:

At the pre-defined interval, AppDaemon internally toggles the `input_boolean.gcal_update`, which triggers the refresh of Google Calendar events, and the HTML files are over-written with new data. However, it will not appear immediately on the Dashboard. Due to the caching issues on HTML, auto-refresh of HTML is done once every minute. Hence, at the most you may have to wait for 1 minute to see the new data after the refresh. If you need to see the events immediately after adding them to Google Calendar, toggle the `input_boolean.gal_update` manually, which will reset to OFF after executing the code.

If you encounter any error or unexpected behavior of gcal_timer app or any other AppDaemon based apps, you could use the following 2 options to determine the source and nature of the error, which may help on debugging.

1. Settings > Add-ons > AddDaemon > Log : Click on REFRESH at the bottom to see any logged info and/or errors. Restarting AddDaemon sometimes clears the error.

2. AddDaemon webserver at http://your_HA_server_ip:5050 > Logs > main_log or error_log.



Log
main_log

Log Entry

```
2023-08-04 17:05:57.671010 INFO gcal_timer: Google Calendar Updated for Ravi
2023-08-04 17:05:31.072418 INFO AppDaemon: Calling initialize() for gcal_timer
2023-08-04 17:05:31.068086 INFO AppDaemon: Loading app gcal_timer using class GCalTimer from module gcal_timer
2023-08-04 17:05:31.054434 INFO AppDaemon: Reloading Module: /config/appdaemon/apps/gcal_timer.py
```