**Preliminary Setup and Notes**

You first need to **establish and create google credentials** before the plugin can access your calendar.

Follow the instructions in the document **“Setting up a Google V3 API Service  
Account”** – when you have finished you should have a credentials file named GCal3.json and a copy of the Calendar ID.

If you are using UI5 and Calendar ID contains any of the following characters you need to substitute the encoded form. This is a limitation of the vera input method in UI5

The two that I know of are:

**& - replace with %26**

**# - replace with %23**

There may be others ....

**Using iCal Calendars**

It is also possible to use **PUBLIC** ical calendars for example (from google using the ICAL link) [https://www.google.com/calendar/ical/en.australian%23holiday%40group.v.calendar.google.com /public/basic.ics](https://www.google.com/calendar/ical/en.australian%23holiday%40group.v.calendar.google.com/public/basic.ics)

or this: <http://sports.yahoo.com/nfl/teams/sfo/ical.ics>

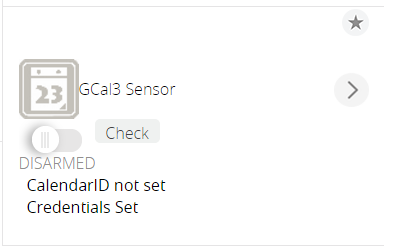
If you are trying the ical feature - the easiest way to tell if you have a 'good' url is to paste it into a browser and see if the browser attempts to download a calendar file for you. Also the url must be in the form of [http:// or](http://or) https://

Also note that ical is inconsistent (allowed by the spec) in the way it stores date and time information. The plugin will correctly handle ical calendars that specify the start/end in utc or an unspecified 'local' time. If it’s unspecified – the plugin will use the time setting from vera. Usually this will be fine so long as the ical sets the events to the same time as the vera expects. If the calendar has a different timezone and does not save the time in utc - then the vera event times will be wrong.

**Installation**

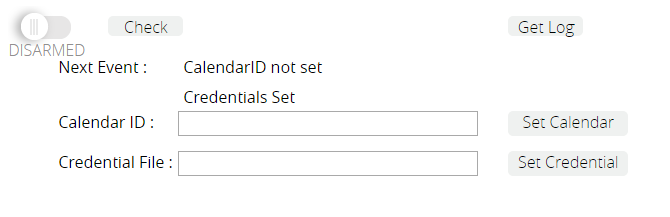
Install from the marketplace. Press F5 on your browser (to refresh). If you get a message that Lua Startup Failed – reboot the vera.

You should then see a front panel that looks like the following. The message lines may read differently if this is a completely new install or if you had a previous version of GCal3 on your vera.



**Initial Configuration**

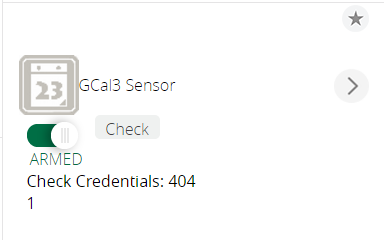
Open the Control tab on the plugin and set the Calendar ID (the one you saved when you did the credentials set-up) and if necessary the Credentials File name (the default – Gcal3.json) is already set in the plugin). Note that Vera will restart (after about 30 sec) with these actions – so wait until this is complete.



**Go to your vera dashboard and upload a copy the GCal3.json file created earlier**

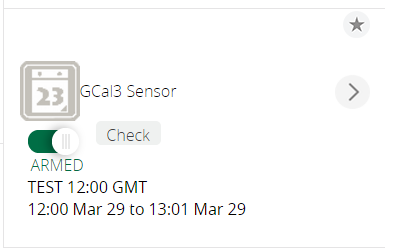
Apps 🡪 Develop Apps 🡪 Luup Files 🡪 Browse [select GCal3.json]. Note that Lua will restart (UI7) if using UI56 then check the restart box

GCal will then check the credentials. If you get a message that looks like this – then your calendar are not set up with the correct permissions or the credential file is incorrect or has the wrong name.

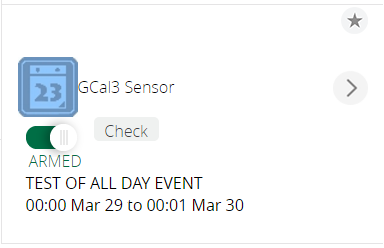


Assuming there are no errors at this stage - Click on “DISARMED” to ARM the plugin.

The front panel of the calendar will look like this: Showing the start and end time of the next event



If it’s an all-day event it will be displayed like this (starts at 00:00 and end at 00:01 the next day).

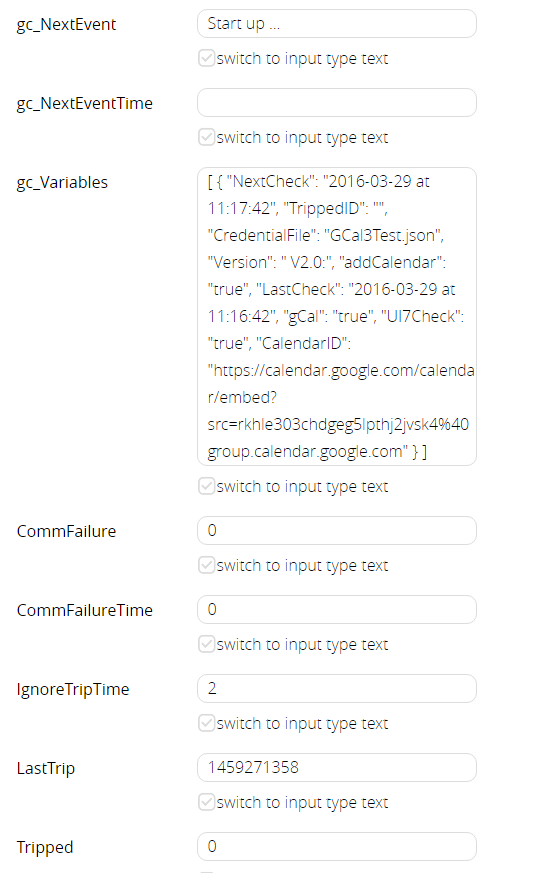


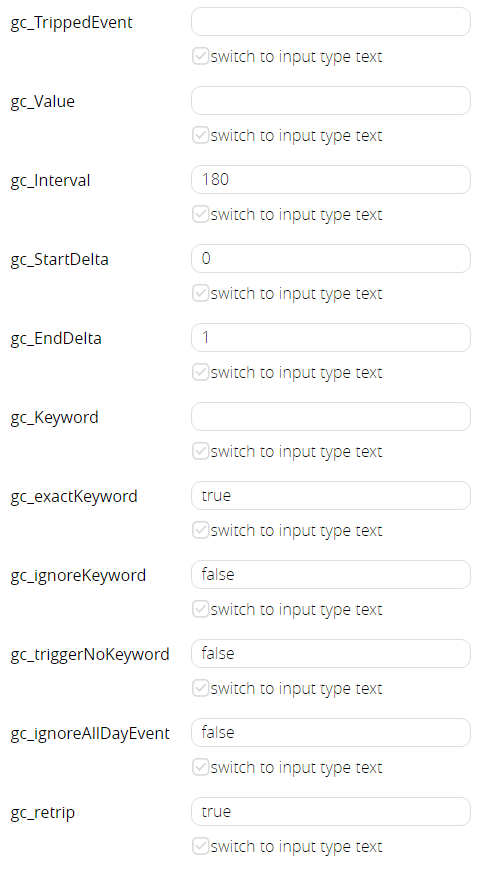
Open the **Calendar Tab** and you should see your calendar. If this is not the case – **the likely cause is that you are not logged into google from your browser.**

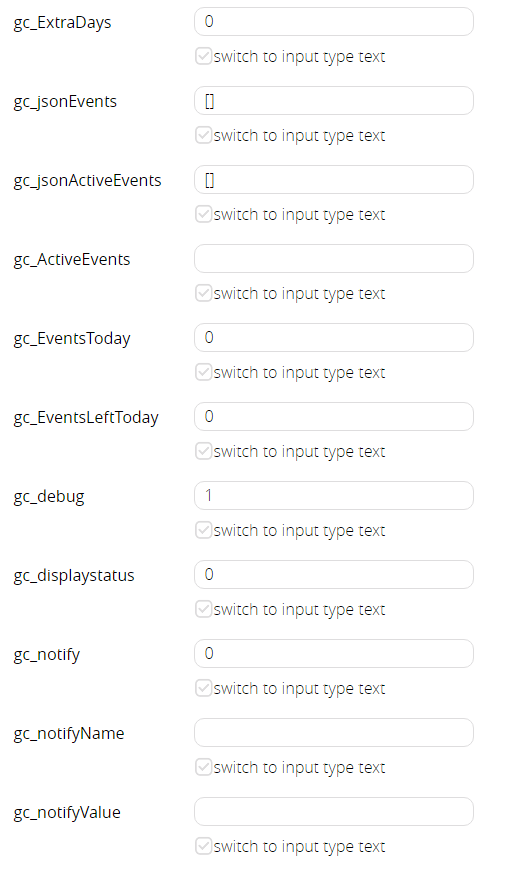
**If there is any error at this stage – you will likely need to look at the log information with gc\_debug (see below) set to 3.**

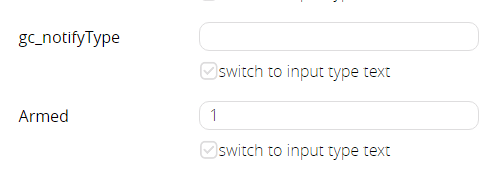
**Configuration Options**

The main way to configure the calendar is through the **variables tab in the advanced tab** – which should now look more or less like this:









Variables in the following table that are in bold are for altering the plugin behavior – all others are either system variables or set by the plugin and are for information.

|  |  |  |
| --- | --- | --- |
| Variable | Value | Comments |
| Tripped | 0 or 1 | **Set by the device.** 0 if not tripped, 1 if tripped |
| gc\_TrippedEvent | An event Name or Blank | **Set by the device.** The name of the currently tripped event |
| gc\_Value | The value of an optional parameter for the tripped event | **Set by the device.** The event name can have optional text in { } at the end. If present it will be placed into this variable as is – without the {} |
| gc\_NextEvent | An event Name | **Set by the device**. The name of the next event that will trip the device or be active.  If the event is tripped – this will be the same as gc\_TrippedEvent**.** |
| gc\_NextEventTime | If an all day event:  All Day -- dd mmm  Otherwise the start and end time  hh:mm mmm dd to hh:mm mmm dd | **Set by the device.** This is the time of the current or next event. If it’s an all day event then only the day and month are shown.  If it’s not an all day event then the start and stop time are displayed including any  allowance for gc\_StartDelta and gc\_endDelta. |
| **gc\_Interval** | A number in minutes  The default is 180 minutes, i.e. 3 hrs. The minimum is 1 minute. | **Configuration.** This is the maximum amount of time  between refreshes of the calendar See notes below. |
| **gc\_StartDelta** | A number in minutes  Default is 0  Negative numbers are preceded by - positive numbers have no + | **Configuration.** This is the number of minutes before (negative) or after the event start time that you want the plugin to turn on. Applies to all events unless an event specific start / end delta is defined (see usage notes below). |
| **gc\_EndDelta** | A number in minutes  Default is 0  Negative numbers are preceded by - positive numbers have no + | **Configuration.** This is the number of minutes before (negative) or after the event end time that you want the plugin to turn off. Applies to all events unless an event specific start / end delta is defined (see usage notes below). |
| **gc\_Keyword** | Can be left blank or can be a keyword or phrase. Multiple keywords can be specified if separated by a semi-colon; | **Configuration.** If set this  specifies the event name and / or description to match. The plugin will force it to upper case.  The behavior depends on the values set for gc\_exactKeyword and gc\_ignoreKeyword |
| **gc\_exactKeyword** | Either ‘true’ or ‘false’ (no quotes)  Default is true | **Configuration.** Set this to true if you want the plugin to match an event name exactly.  Set to false if you want the  plugin to match anywhere in the event name. |
| **gc\_ignoreKeyword** | Either ‘true’ or ‘false’ (no quotes)  Default is false | **Configuration.** Set this to true if you want the plugin to ignore any events where the event name matches the keyword or the description contains the keyword |
| **gc\_triggerNoKeyword** | Either ‘true’ or ‘false’ (no quotes)  Default is false | **Configuration.** Set this to true if you want the plugin to trigger the plugin if no keywords are  set. This means the plugin will trigger on every event. It will usually be better to set to false and use the “An event has specified name” option. |
| **gc\_ignoreAllDayEvent** | Either ‘true’ or ‘false’ (no quotes)  Default is false | **Configuration.** Set this to true if you want the plugin to ignore all day events. |
| **gc\_retrip** | Either ‘true’ or ‘false’ (no quotes)  Default is true | **Configuration**. Set this to false so that back-to-back events with the same name will be treated as one long event. Useful for  repeating all-day events. |
| **gc\_ExtraDays** | A number in days.  Default is 0 | **Configuration.** Usually GCal looks for events up to 1 day into the future. In some cases a user needs to know further in advance – this variable adds extra days to the default of 1 day. |
| gc\_jsonEvents | Json encoded array of pending valid events | **Set by the device.** See notes below. |
| gc\_jsonActiveEvents | Json encoded array of active events and parameters i.e. current nested and overlapping at the current time | **Set by the device.** See notes below. |
| gc\_ActiveEvents | Comma delimited list of active events i.e. current nested and overlapping at the current time | **Set by the device.** See notes below. |
| gc\_EventsToday | Gives a count of the number of events present in the calendar in the current (local time) 24 hrs. This will remain the same during the day. | **Set by the device.** |
| gc\_ActiveEventsToday | Gives a count of the number of events remaining in the current (local time) 24 hrs.  This will start out equal to gc\_EventsToday and decrease as each event completes. When no more events remain it will equal 0 | **Set by the device.** |
| gc\_lastCheck | The date and time the plugin last checked the calendar | **Set by the device.** |
| gc\_nextCheck | The date and time the plugin will next check the calendar | **Set by the device.** |
| **gc\_debug** | A value from 1 to 3 Default is 1 | **Configuration**. The number specifies the debug level. 1 is lowest and recommended for everyday use. 2 provides more information and 3 is used for troubleshooting. |
| gc\_displaystatus | A value or 0 when there is not tripped or active event, 50 when an active event and 100 when a tripped event. Varies the icon on the plugin panel | **Set by the device.** |
| gc\_notify | Set to "1" at the Start and End of an event.  15 sec later is reset to 0 and other variables (below) are reset to "" | **Set by the device.**  Works on event start and end - including overlapping events. The events must still be 'eligible'.  If events have the same start or end time - the notifications are spaced approximately 5 seconds apart. |
| gc\_notifyName | The event name when gc\_notify is “1” | **Set by the device.** |
| gc\_notifyValue | Any event parameter (i.e. the bit in {}) if gc\_notify is “1” | **Set by the device.** |
| gc\_notifyType | Either "Start" or "End"  If gc\_notify is “1” | **Set by the device.** |

**Usage Notes**

1. Events can be named in the calendar as a word or words (**do not use quotes**) e.g. “Occupied” or “Wake Up”
2. Descriptions can be used in the calendar as if they are events.
3. Zero or more keywords can be defined in the plugin (gc\_keyword), if more than one keyword is defined, they are separated by a semi-colon i.e. ;
4. Events, description and keywords are case insensitive -- so 'Test Event' = 'test event' = 'TeSt EVeNT' White apace is trimmed from the start and end and they are forced to upper case - this just makes it a little more robust to typo's
5. If no keyword is set then any event will enable the plugin to be used to activate a scene. This can be done in two ways depending on the value of gc\_triggerNoKeyword. When set to true the plugin is ‘triggered’ for every event. If set to false (recommended) then the scene can be specified to react to “An event has specified name” this does not technically trigger the plugin – but the event is ‘active’.
6. If keyword is set and gc\_exactKeyword is true. Then an event name or description must match the keyword exactly – there can be nothing before and nothing after **except for options** (see below).
7. If keyword is set and gc\_exactKeyword is false. Then the keyword can match anywhere in the event name or the description.
8. If a keyword is set and gc\_ignoreKeyword is false then only those events that match the keyword can trigger the plugin.
9. If a keyword is set and gc\_ignoreKeyword is true then any events that match the keyword are ignored – it’s as if they were not in the calendar in the first place.
10. If gc\_ignoreAllDayEvent is true then all day events are ignored – it’s as if they were not in the calendar in the first place.
11. There are two **optional** parameters that can be used in defining keywords (in the plugin), events and / or descriptions (in the calendar). These are : **[x,y]** and **{some parameters}**[x,y] specifies a start(x) and end(y) delta (in minutes)  
    {some parameter } is used to populate gc\_Value and can be used by other plugins as input.   
    While event names, descriptions and keywords are all forced to upper case anything in {} is left as-is.
12. There is a strict order of precedence for these optional parameters – meaning that if [] or {} are used, then the first time they are encountered is used andy subsequent uses (for an event) are ignored. The order is:  
    Keyword -- if specified for a plugin keyword, this over-rides anything below in precedence  
    Event -- if specified in the event, this over-rides anything below in precedence  
    Description -- if specified in the description, this over-rides anything below in precedence  
    Default --- the default for [x,y] is [gc\_startDelta,gc\_endDelta] and for {} is “” i.e. empty string

The optional parameters can be used in any order and different levels of precedence – although it’s probably simpler and easier to be sure of what is happening if you use one level

Two examples:

**Example 1 - No parameters**

An all day event "All Day"

An event "A keyword" on the same day

An event "Not a keyword" on the same day

and I have the following:

gc\_Keyword = a keyword

gc\_exactKeyword = true

gc\_ignoreKeyword = true

gc\_ignoreAllDayEvent = true

The plugin will trigger on the event "Not a Keyword" -- because all day events and any events with the keyword do not (logically) exist because gc\_ignoreKeyword = true and gc\_ignoreAllDayEvent = true

If I change gc\_ignoreAllDayEvent = false then the plugin would trigger on "All Day" because it's as if events with the keyword do not (logically) exist

If I next change gc\_ignoreKeyword = false then it would trigger on "A Keyword" -- i.e. we are back to the 'default' behavior

Lastly with gc\_ignoreKeyword = false and gc\_ignoreAllDayEvent = true then it will trigger on the event "A Keyword" (but if there was an All Day event called "A Keyword" it would be ignored)

In the above example gc\_exactKeyword was true. If I now change gc\_exactKeyword to false and gc\_Keyword to “keyword” then the two events "A keyword" and "Not a keyword" will be treated as if they satisfied the keyword and would trigger the plugin (or not) based on the setting of gc\_ignoreKeyword.

**Example 2 – using [] Parameters**

Lets say I have gc\_startDelta = -5 and gc\_endDelta = 5

I have a keyword = This Keyword [-10,10] I have 3 events

1. This Keyword [7,7]
2. This event [8,8]
3. Another event

The effective events would be (respectively)

1. This Keyword [10,10] i.e. because it matches the keyword the keyword parameter takes precedence
2. This event [8,8] i.e. it’s not matched by the keyword so it’s left alone
3. Another event [-5,5] i.e. defaults

Note that this example does not take into account the trigger conditions.

**The variable gc\_jsaonEvents**

The variable gc\_jsonEvents is populated with a summary of the eligible events for use by other plugins / scenes. gc\_jsonEvents is json encoded with each entry comprising the effective start time, end time and name of the event. Note that the start and end times are **local times** (i.e. the timezone of your vera) - *so you should not need to do any timezone manipulation*.

To help illustrate - here is some test code I use to read gc\_jsonEvents

local function getjsonEvents() -- this is really some sample code and useful for debugging

*function testJson()*

*local GCAL\_SID = "urn:srs-com:serviceId:GCalIII"*

*local jsonEvents = luup.variable\_get(GCAL\_SID, "gc\_jsonEvents",lul\_device)*

*if (jsonEvents == "[]") then -- equivalent of a nul so don't try*

*return*

*end*

*local json = require(“json”)*

*local eventList =json.decode(jsonEvents)*

*package.loaded.json = nil*

*local numberEvents = #eventList*

*local startevent, startDate, startTime, endevent, endTime, eventname, event*

*for i = 1,numberEvents do*

*startevent = eventList[i].eventStart*

*startDate = os.date("%Y-%m-%d", startevent)*

*startTime = os.date("%H:%M:%S", startevent)*

*endevent = eventList[i].eventEnd*

*endTime = os.date("%H:%M:%S", endevent)*

*eventname = eventList[i].eventName*

*end*

*return*

*end*

Similarly the variable gc\_jsonActiveEvents gives a json array of the currently active events – not just the first event but any overlapping or nested events. The event name and any associated parameter are provided. This variable can be used in the same manner as described above by Luup/Lua code to trigger scenes.

For simpler uses, gc\_ActiveEvents is a comma delimited list of active/overlapping/ nested events (with no parameter) that can be used by Luup/Lua/PLEG to trigger scenes.

**Interval Notes**

The variable gc\_Interval (in minutes) sets the maximum time between checks of the calendar. The plugin will check the calendar at the start and end of each 'valid' event (i.e. an event that will cause it to trigger). Also - if the time between the end of one valid event and the next valid event is more than gc\_Interval, then it will check gc\_Interval minutes after the first event and so on until the start of the next event. So if there are no events in the near future it will check every gc\_Interval minutes to see if anything has changed.

Each check of the calendar looks at a 24 hr window (+ gc\_Interval) from 'now' (negative startDelta and positive endDelta will alter this a bit). So events more than 24 hrs from 'now' (+ gc\_Interval) will not be seen.

The default value for gc\_Interval is 3 hrs as a compromise between 'annoying' google (and vera) too often and seeing new events in the calendar. If you place a new event in the calendar and it's less than 3 hrs from now - it may not be seen unless the plugin had a check scheduled before the new event (either another event or just lucky). So - if you NEEDED a new event to be seen and it was scheduled in the next 3 hrs (or gc\_Interval minutes if you use something other than the default) you'd need to press the check button on the plugin to force an update.

Setting gc\_intterval to a large number (like 24hrs - or one week) runs the risk of not seeing newly entered events). Setting it less than the default is likely a waste of vera resources (and maybe google will get upset with you if it's really small )

**General Notes**

For ‘eligible events (i.e. (taking into account gc\_keyword, gc\_exactKeyword, gc\_ignoreKeyword and gc\_ignoreAllDayEvent)

1. If events overlap – then the first event will complete before the second event starts i.e. the second event will have an effective start time at the end of the first event.
2. If an event is completely ‘enclosed’ by another event – then it will not be seen.
3. If two events have the same start time – it is undefined as to which one will trigger the plugin.
4. If there are two events with the same event name and the second event has the same start time as the end of the first – they are viewed as one continuous event. i.e. the plugin does not ‘untrigger’ then ‘retrigger’.
5. The minimum duration for an event is 1 minute taking into account the effect of any start and end delta. If the effective duration is less than one minute then the event will not be seen by the plugin i.e. it will be discarded.