WEATHER STATE & FORECAST SUITE USER MANUAL

Version 2.8.1 standalone

TABLE OF CONTENT

1. PREFACE

1.1 PLEASE DONATE TO SUPPORT MY WORK

Dear user, if you find my work useful to you then please support my work with donation. This will help me keep making more solutions for your home automation. You can donate any amount of money that you see fit by PayPal on this link 

or you can contact me on Fibaro forum message Inbox or directly on my e-mail: z.sankovic@gmali.com for any other type of the support. THANK YOU! 😊

1.2 LEGAL NOTICE

Fibaro and **Home Center 2 (HC2)** are registered trademarks of **Fibar Group S.A.** registered in Poland and other countries.

1.3 TERMS AND CONDITIONS

Permission to use, copy, modify, and distribute this software and its documentation for educational, research, personal use and non-profit purposes, without fee and without a signed licensing agreement, is hereby granted, provided that the above copyright notice, with "Terms and Conditions" and "Disclaimer" appear in all copies, modifications, and distributions.

It is strictly forbidden to sell, rent, lease and/or lend this software for profit without prior consent from the Author. Please contact the Author by e-mail: z.sankovic@gmail.com for commercial licensing opportunities.

1.4 DISCLAIMER

This software is provided by copyright owner "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the author and distributor be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

Fibar Group S.A. and their employees **are not** responsible for support of the Weather State & Forecast module in any way. Please contact the author, on the Fibaro Forum, for any questions or support required. The Author however, wishes to acknowledge the help received from the Fibar Group S.A. in that they have kindly supplied the use of a HC2 for development purposes, free of charge.

2. INTRODUCTION

Weather state & forecast VD enhance HC2 weather capabilities. Weather state and forecast module in its latest version works with following most popular weather services:

- OpenWeatherMap – providing current weather 6 day hourly and daily forecast
- AccuWeather – providing current weather and 5 day / 12 hourly forecast
- WeatherBit – providing current weather and 16 day daily forecast
- Weather HERE – providing current weather and 7 day daily forecast
- Weather API – providing current weather and free 3 day daily forecast
- Weather Unlocked – providing current weather and 7 day daily forecast
- Wunderground PWS – providing PWS measurement and 5 day daily forecast

2.1 PREREQUISITES

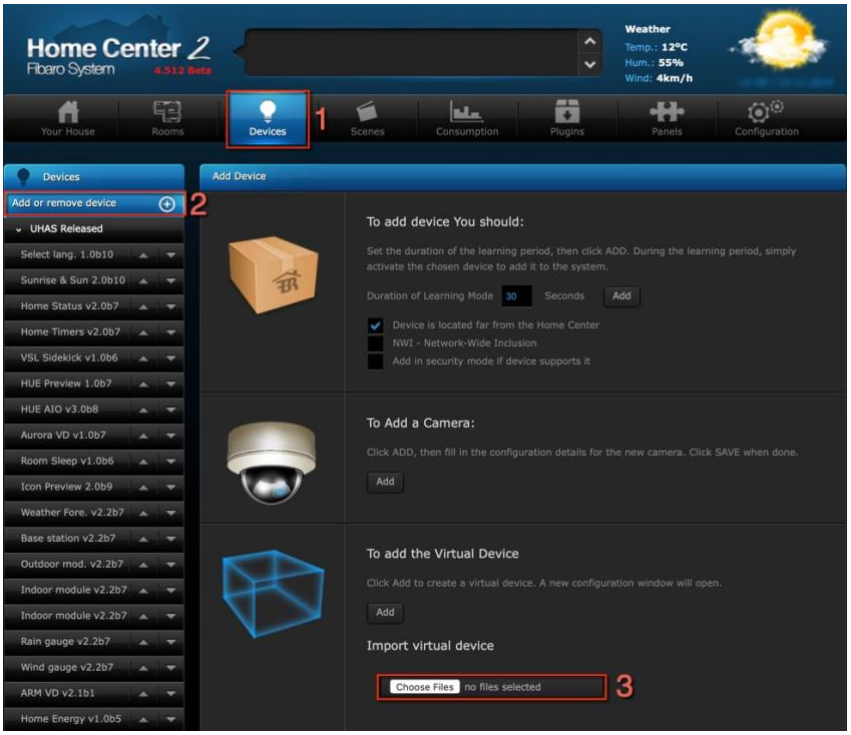
- Fibaro Home Center 2 with firmware 4.600 or greater with installed **Weather provider plugin** set as **main** weather provider.
- Sankotronic Lab. **Emoji VD** standalone
- User account on at least one of the available weather forecast providers with API key

2.2 FEATURES

- Works with 7 most popular weather services on the planet
- Regularly refreshes weather state and forecast
- User can setup when forecast push and popup notifications will be sent same as e-mail with complete forecast
- Can install more than one set of VD's and follow weather for more than one place
- Can display Netatmo measurements for temperature, humidity, wind and precipitation
- User can add additional sensor for lux and UV
- Automatically finds weather provider plugin if installed and can then update weather on the HC2 web GUI and mobile applications (Weather provider plugin must be set as default)
- VD has built in multi-language support with 27 languages included (see APPENDIX 1 for the list). VD can use HC selected language or user can select any other available language. VD will be automatically translate current weather and forecast data to selected language (not all weather services support all 27 languages. Please check API documentation for selected weather service)
- Easy setup for VD icons. User can download HC2 Icon Preview VD to easily find icon ID numbers
- Global variables are automatically added, monitored and repaired and does not require user intervention
- Supported are metric and imperial units for measured data. Wind speed can be set to show in meters per second (m/s) instead of km/h or mph

3. WEATHER STATE VD INSTALLATION AND SETUP

Weather State VD should be imported first into HC2 by choosing "Add or remove device" on the devices page by choosing the option to import a Virtual Device:



After importing VD click on Advanced tab, scroll all the way down to main loop code window and click on Debug button. Debug window will open and this debugging messages will be shown:

```
[DEBUG] 14:55:08: Standalone Weather State VD version 2.8 - (c) 2018-21 Sankotronic
[DEBUG] 14:55:08: [13.04.21]: No language selected. Will use HC default language
[DEBUG] 14:55:08: [13.04.21]: Global variable WeatherFore created successfully
[DEBUG] 14:55:08: [13.04.21]: Global variable WeatherFore check OK
[DEBUG] 14:55:08: [13.04.21]: stationNo: 1 forecast set of values for this VD already exists
[DEBUG] 14:55:09: [13.04.21]: Global variable WeatherTable created successfully
[DEBUG] 14:55:09: [13.04.21]: Global variable WeatherTable check OK
[DEBUG] 14:55:09: [13.04.21]: Global variable WeatherTable updated successfully
[DEBUG] 14:55:10: [13.04.21]: Global variable WeatherMain_53 created successfully
[DEBUG] 14:55:10: [13.04.21]: Global variable WeatherMain_53 check OK
[DEBUG] 14:55:10: [13.04.21]: Global variable WeatherMain created successfully
[DEBUG] 14:55:10: [13.04.21]: Global variable WeatherMain check OK
[DEBUG] 14:55:11: [13.04.21]: Global variable WeatherLang created successfully
[DEBUG] 14:55:11: [13.04.21]: Global variable WeatherLang check OK
[DEBUG] 14:55:11: [13.04.21]: Global variable WeatherLang updated successfully
[DEBUG] 14:55:11: [13.04.21]: Global variable WeatherState created successfully
[DEBUG] 14:55:11: [13.04.21]: Global variable WeatherState check OK
[DEBUG] 14:55:11: [13.04.21]: stationNo: 1 weather set of values for this VD already exists
[DEBUG] 14:55:11: [13.04.21]: Checking VD icons
[DEBUG] 14:55:11: [13.04.21]: WARNING Missing AccuWeather apiKey to get location key
[DEBUG] 14:55:11: [13.04.21]: VD Updated
```

As soon as VD is imported it will add all necessary global variables and then activate update button.

User can also check debug window of the Update  button that should show following debugging messages:

```
[DEBUG] 15:41:07: [13.04.21]: WARNING Didn't find Weather Provider plugin. Please install and set as default
[DEBUG] 15:41:07: [13.04.21]: ERROR reading data from weather service
[DEBUG] 15:41:07: [13.04.21]: WARNING Forecast VD not found. Please install
```

Errors shown in Update button debug window are normal since user first must setup weather service to enable VD to work. For that user will need to open account and get API key.

To get API KEY just follow online instructions. Links to the services are provided in introduction part of this manual.

If first line of debug windows is as on above picture then user need to install Weather provider plugin:

and set it as default weather provider in HC2 Configuration settings so that Weather State VD can also update weather of the GUI interface and mobile applications:

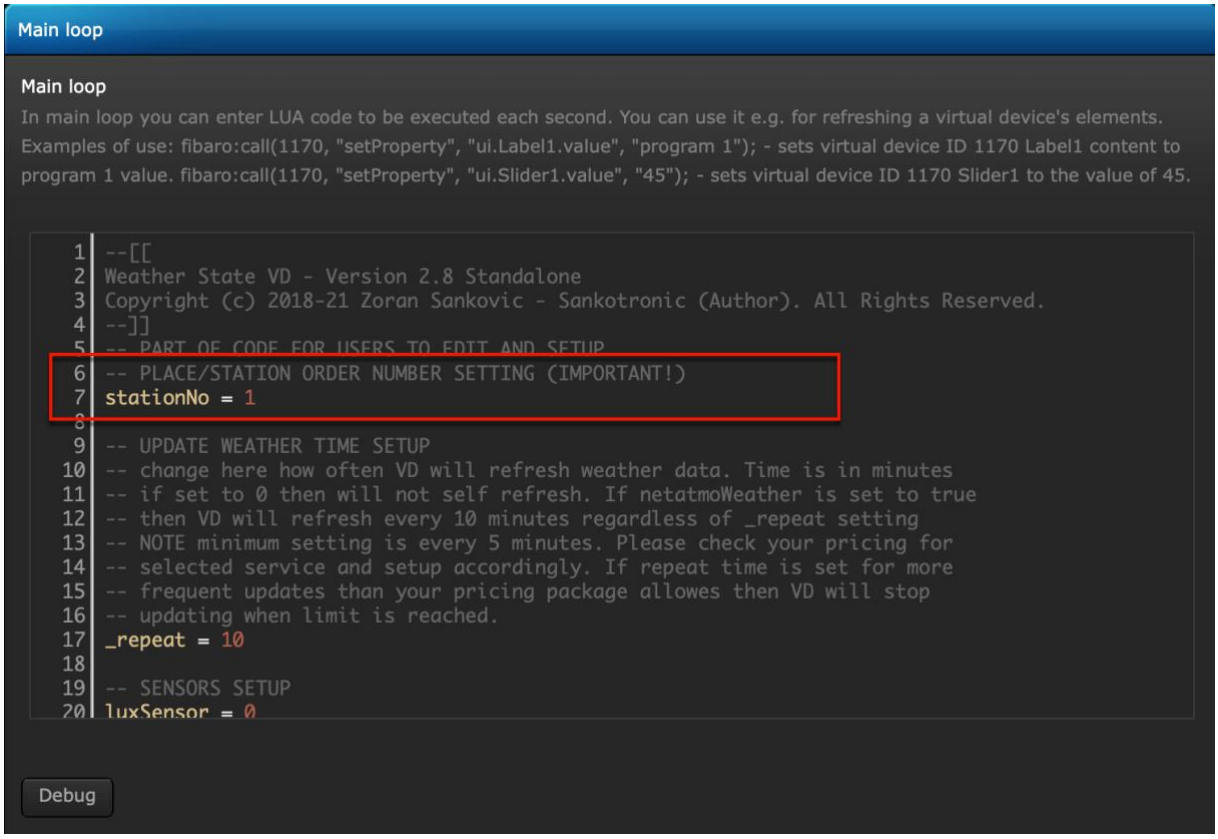
To install Weather provider plugin user can go to Plugins section and under Available Climate section can find this plugin. If Weather provider plugin is properly installed then Update button debug window will show:

```
[DEBUG] 15:42:32: [13.04.21]: Found Weather provider plugin ID: 55
[DEBUG] 15:42:32: [13.04.21]: ERROR reading data from weather service
[DEBUG] 15:42:32: [13.04.21]: WARNING Forecast VD not found. Please install
```

NOTE – Weather State VD will be referred as WS VD and Weather Forecast VD will be referred as WF VD further in this manual.

3.1 STEP 1 – WS VD SETTING STATION NUMBER

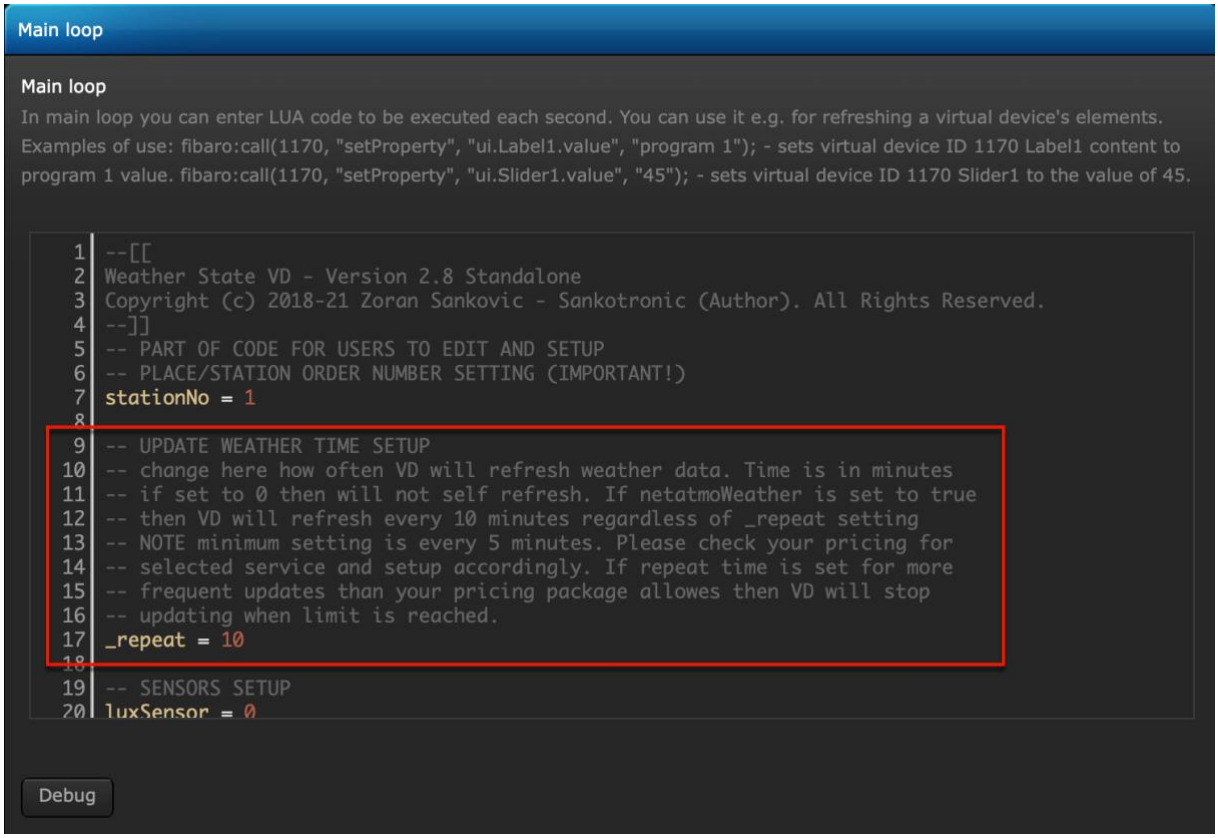
WS VD and WF VD can be imported multiple times if user wants to have weather reports for more than one station. Setting for station number is in VD main loop. First imported VD station number **stationNo** should be left set to 1. For each additional imported WS VD that number should be increased for +1:



IMPORTANT NOTE – Regardless of how many WS VD and WF VD user import they will all use one scene. WS VD checks if scene is running and waits until scene stops and then starts the scene. WS VD also waits for scene to finish fetching data from weather provider server and then continues with updating itself and WF VD for that station.

3.2 STEP 2 – WS VD SETTING REPEATING TIME FOR UPDATE FREQUENCY

Default setting for updating weather state is set to every 10. If user need to change how often WS VD will update then can do that in VD main loop code:



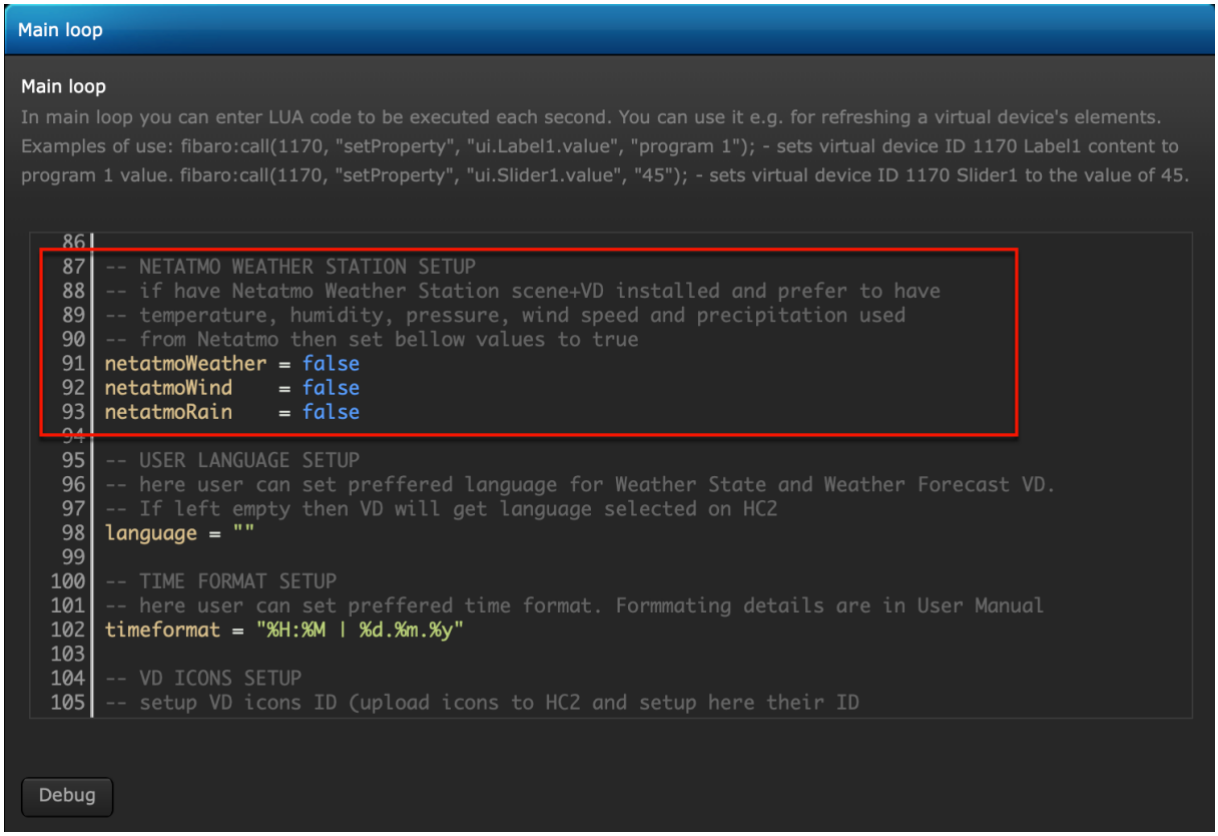
WARNING – Setting **_repeat** time bellow 5 minutes is not possible because it could exceed weather service provider free account quota! If user set repeating time bellow 5 minutes there will be a message in main loop debug window:

Please consult selected weather service website and calculate update frequency depending on number of installed WS VD's and available calls to the service. If SW VD still exceeds available quota then it will continue to report errors till the end of that day. Solution is to increase repeating time and wait until next day for error to disappear.

NOTE – if **_repeat** is set to **0** then WF VD will not update itself and user must have his/her own code to press Update button e.g. call from Main scene FTBE:

3.3 STEP 3 – WS VD SETTING NETATMO WEATHER

If user have Netatmo Weather Station and have installed Netatmo Weather Station suite v3.3 from Sankotronic LAB™ and want to show readings from Netatmo Weather station for temperature, humidity, pressure, wind and rain then in main loop set **netatmoWeather** to **true**:

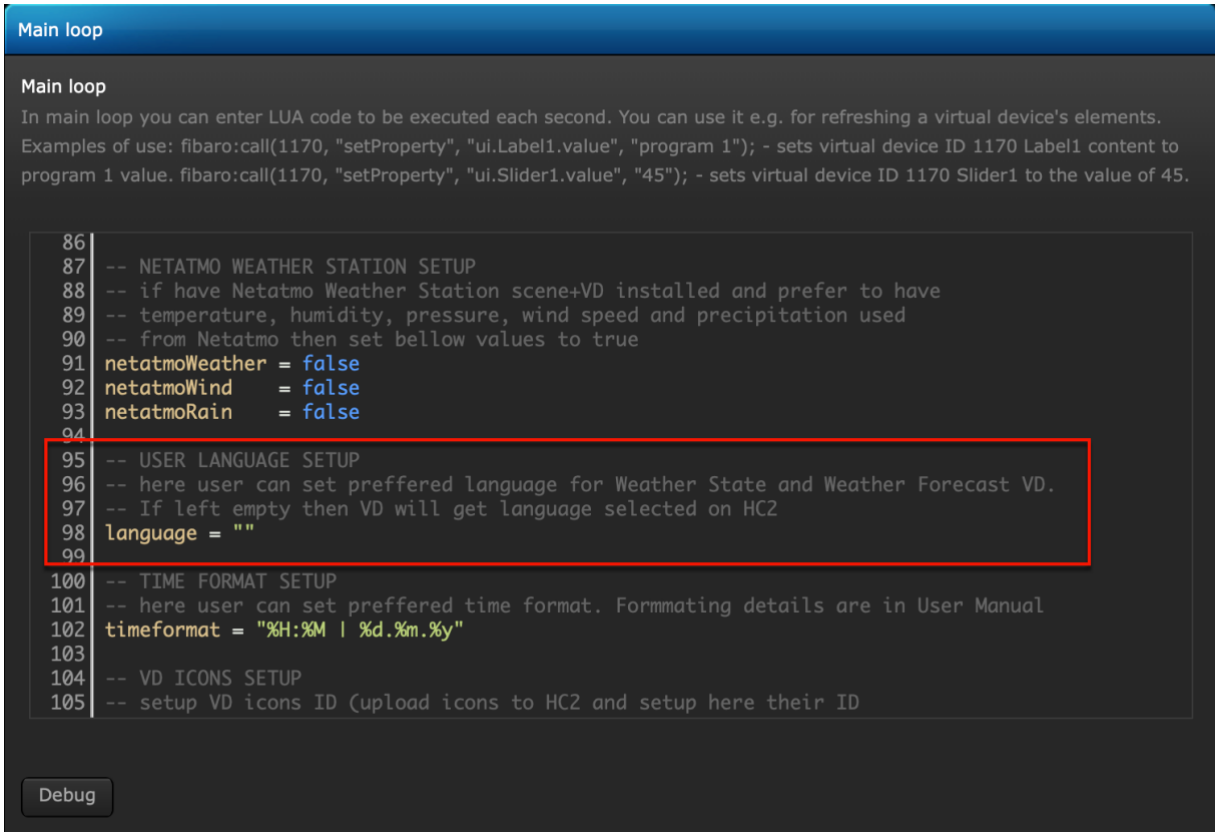


Also if user have Netatmo rain and wind gauges then can set **netatmoWind** and **netatmoRain** to **true** to show their data on Weather State VD.

NOTE – Netatmo Weather Station suite older than v3.3 does not support this option.

3.4 STEP 4 – WS VD SETTING PREFERRED LANGUAGE

WS VD is distributed in English language by default. When first time imported, WS VD will check default language that is selected on Home Center 2 and will translate to that language. **Please refresh browser page after importing WS VD to see translation before making any changes and saving VD**. The user can choose a different language, by opening the WS VD for editing and scrolling down to VD main loop code window. Scroll code until this part is shown:



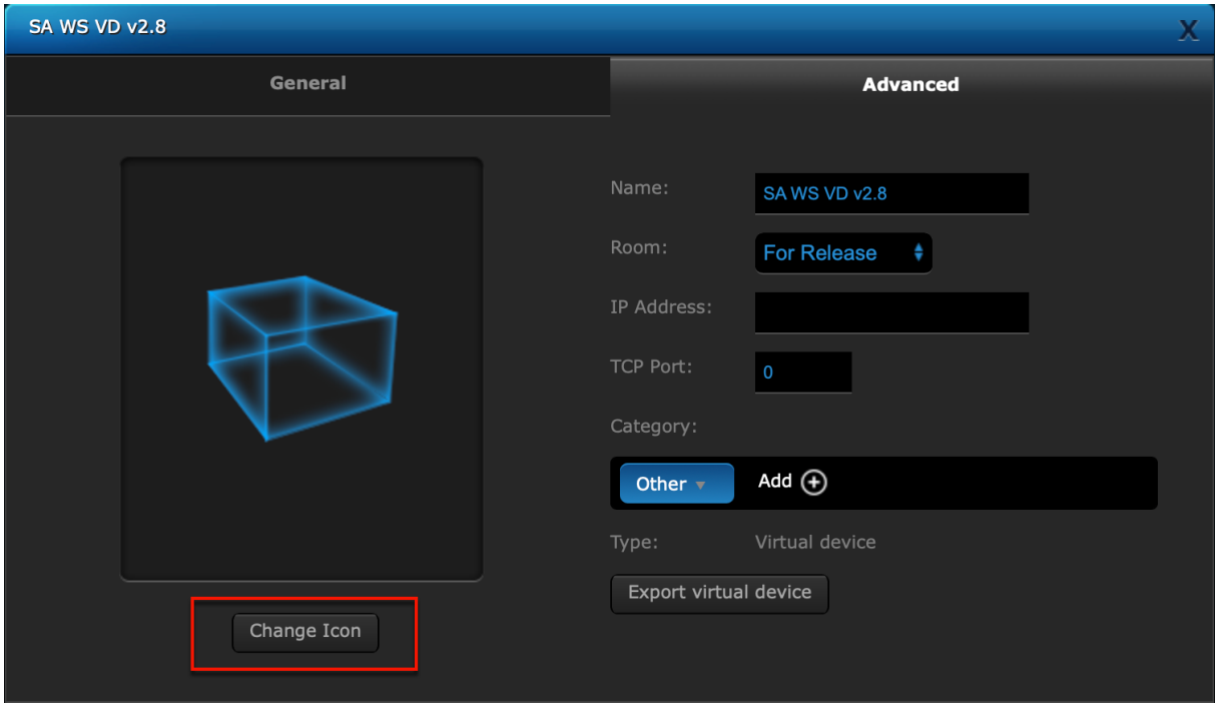
Between quotation marks user can enter two letter code for desired language for WS and WF VD. Two letter language codes can be found in this manual APPENDIX 1.

After changing VD language and saving it WS VD and WF VD will update weather in selected language.

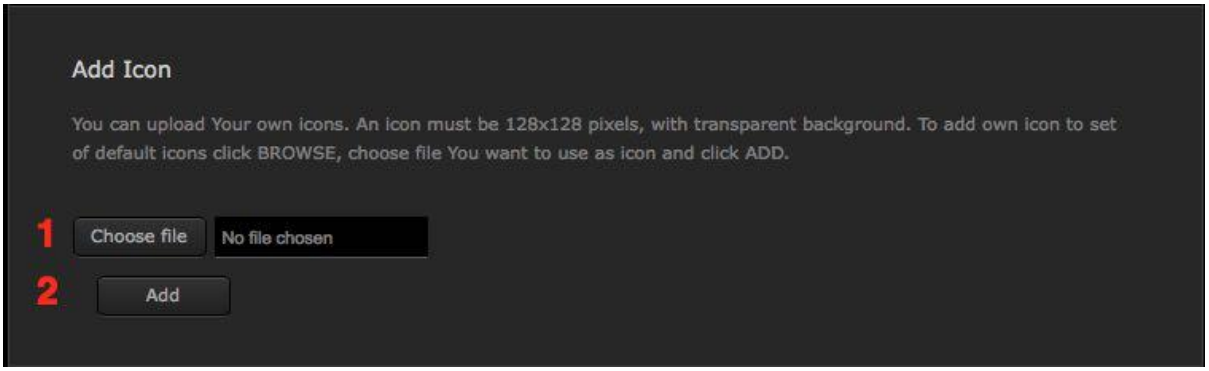
NOTE - It is possible that weather service provider does not support selected language. In that case weather will be received in “en” English language.

3.5 STEP 5 – WS VD ICONS SETUP

User should first import icons to Fibaro Home Center 2. To do that open WS VD for editing and then click on Change Icon button located bellow main icon frame:



New window will open:



First click on **Choose file** button. In new opened window browse to the folder where are extracted files from Weather State & Forecast suite zip file. Select first icon named ***01 - Main_icon 3200.png*** and then click on **Add** button.

Repeat adding other 41 icons in the following order:

02 – clear day 32.png
03 – clear night 31.png
04 – broken clouds day 28.png

05 – broken clouds night 27.png
06 – few clouds day 34.png
07 – few clouds night 33.png
08 – scattered clouds day 30.png
09 – scattered clouds night 29.png
10 – overcast 26.png
11 – fog 44.png
12 – haze 20.png
13 – hail light 17.png
14 – hail 35.png
15 – drizzle 9.png
16 – light rain 10.png
17 – rain 11.png
18 – heavy rain 40.png
19 – sleet 18.png
20 – shower sleet 6.png
21 – flurries 13.png
22 – light snow 16.png
23 – snow 42.png
24 – heavy snow 43.png
25 – light thunderstorm 2.png
26 – thunderstorm 4.png
27 – heavy thunderstorm 1.png
28 – ragged thunderstorm 3.png
29 – windy 23.png
30 – gale 24.png
31 – hot 36.png
32 – cold 25.png
33 – tornado 0.png
34 – chance rain day 50.png
35 – chance rain night 51.png
36 – chance sleet day 52.png
37 – chance sleet night 53.png
38 – chance snow day 54.png
39 – chance snow night 55.png
40 – chance thunderstorm day 56.png
41 – chance thunderstorm night 57.png
42 – error_icon.png

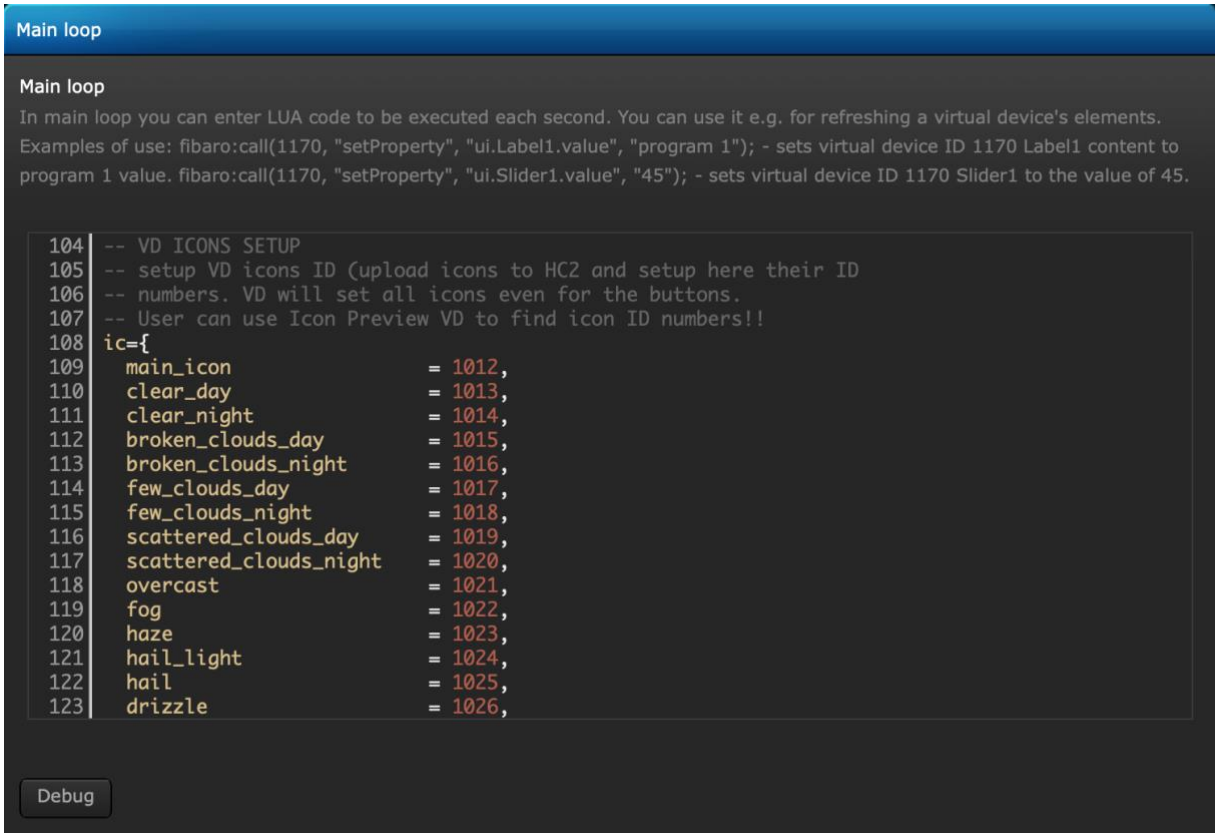
This order is important because after importing the user have to find out only icon ID number for the first one imported and the rest will be increased by one.

When all icons are imported then right click on the first one "***01 - Main_icon 3200***" and from menu select option **Open Image in new Tab** in Safari and Chrome, **View Image** in Firefox, and **Inspect element** in MS Edge.

In new opened tab or inspection window the name of the icon which contains ID number is shown e.g. **User1020.png**. Remember that number. Do not select icon to close add icons window but exit by clicking Devices on main menu.

It is also possible to use HC2 Icon preview VD from Sankotronic LAB™ to find out imported custom icon ID numbers.

Open WS VD for editing and scroll down on advanced tab to VD main loop code window and scroll code until this settings:

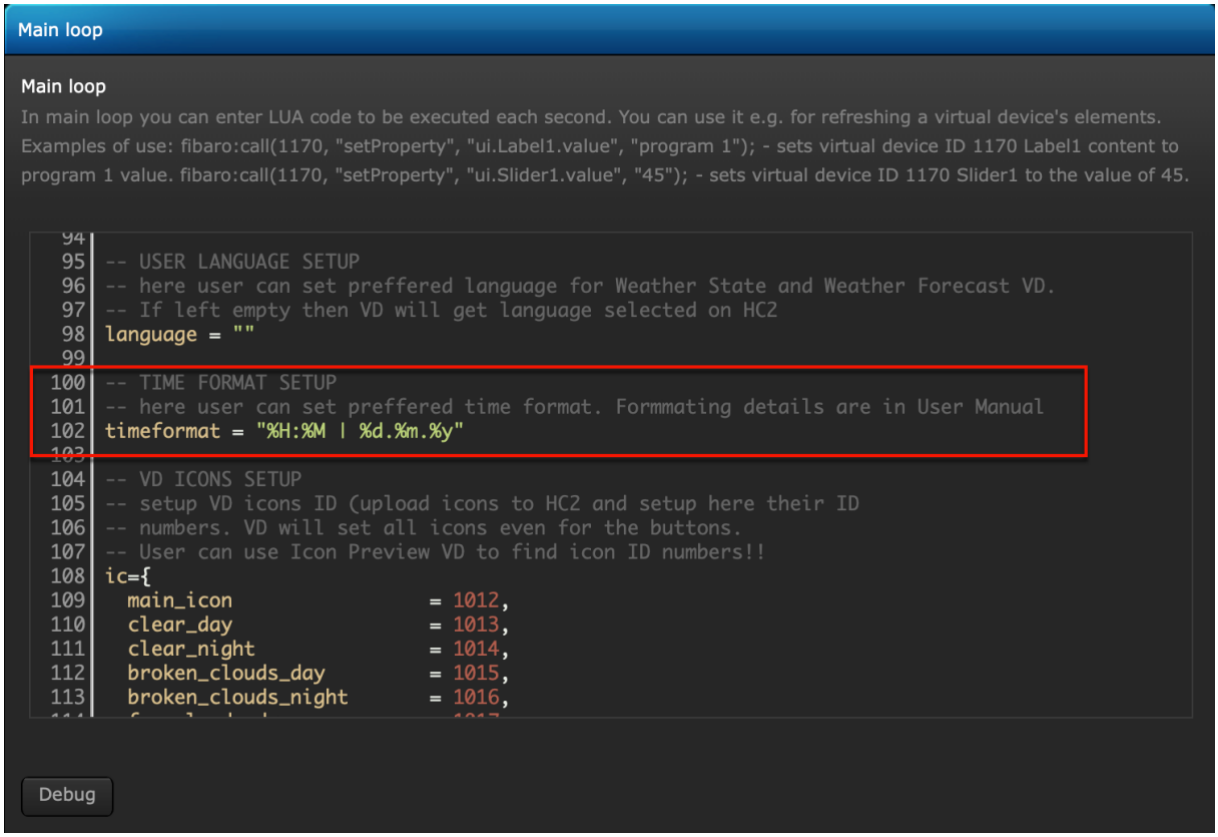


Replace number 1 with icon ID's that are imported. If the icons are imported by the order suggested, then it will be easy to enter them in this settings one by one. Please do not overwrite or delete comma after the number.

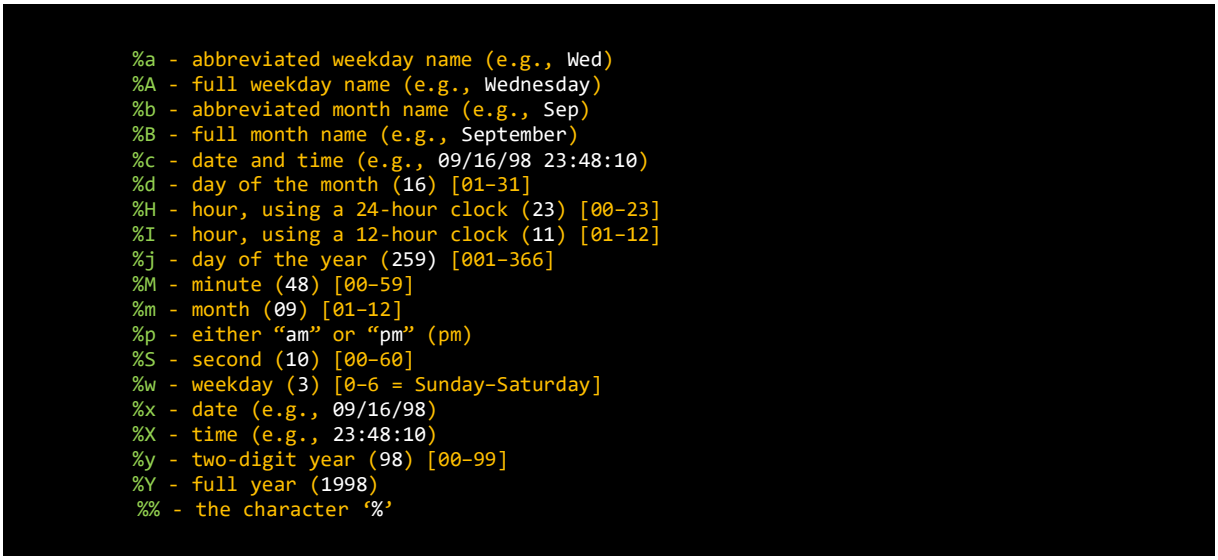
After saving VD, refresh page and all buttons will also have icon set.

3.6 STEP 6 – WS VD SETTING TIME FORMAT

Go to VD main loop code and scroll code until this settings are shown:

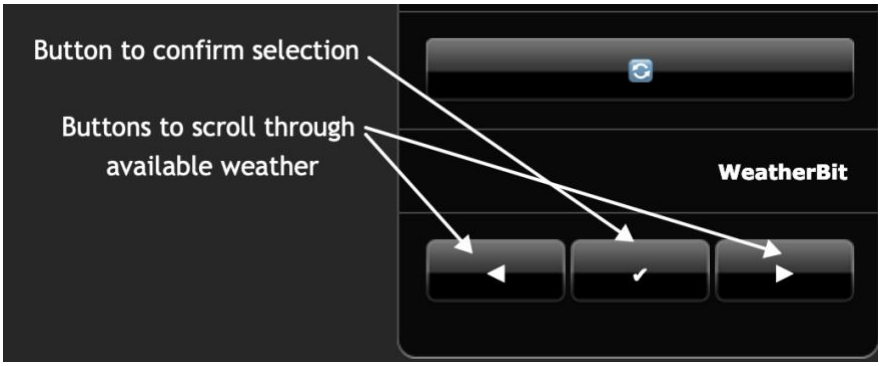


Here user can define how time and date will be shown on WS VD. User can use following table to format time and date:



3.7 STEP 7 – WS VD SETTING WEATHER SOURCE

Latest version of WS VD provide buttons on the bottom of the VD to select and activate preferred weather service:



Use buttons with arrows to select preferred weather service provider and then use middle button to confirm selection. If service is not properly setup with API key then for a brief moment there will be red cross after the weather service provider name:

That means either that Weather Service scene is not installed or was installed before WS VD in which case user need to install scene so that scene can find all installed WS VD and give them scene ID. It is also possible that selected weather service is not properly setup with personal data e.g. API key.

If everything is properly setup then after confirming selection of weather service a green checkmark will briefly show:

And VD will first run scene to get weather data and then show current weather and also update WF VD.

NOTE - If user setup all weather services with API keys then will be able to change weather service on the go with mobile device. This is big advantage since if any of the services stop working user can always select another service at any time and any place.

3.8 STEP 8 – WS VD SETTING WEATHER PROVIDER API KEY AND LOCATION

All weather services setup is located in the VD main loop:

```
-- OPENWEATHERMAP PERSONAL SETTINGS
-- free 1000 calls/day. Recommended minimum repeat every 5 minutes because
-- Weather State uses 2 calls (current & forecast) per one update
-- if no city is setup then will use HC2 location coordinates
cityID  = "[redacted]"
city    = ""
country = "hr"
opn_key = "[redacted]"

-- ACCUWEATHER PERSONAL SETTINGS
-- free 50 calls/day. Recommended minimum repeat every 60 minutes because
-- Weather State uses 2 calls (current & forecast) per one update
-- only HC2 location or lat/lon coordinates are used
accu_apiKey = "[redacted]"

-- WEATHERBIT PERSONAL SETTINGS
-- if no city is setup then will use HC2 location coordinates
wbt_city    = ""
wbt_country = ""
wbt_key     = "[redacted]"

-- WEATHERHERE PERSONAL SETTINGS
-- free 250K calls per month in total. Uses separate call for
-- current weather and for forecast
-- if no city is setup then will use HC2 location coordinates
whr_city        = "[redacted]"
whr_country     = "HR"
-- Depreciated - use only if you still have them else get apiKey
whr_appID       = "[redacted]"
whr_appCode     = "[redacted]"
-- NEW possible to get from developer.here.com
whr_apiKey      = "[redacted]"

-- WEATHER API PERSONAL SETTINGS
-- free 1000K calls per month in total. Uses one call for both
-- current and forecast weather
-- if no city is setup then will use HC2 location coordinates
wap_city = ""
wap_key  = "[redacted]"

-- WEATHER UNLOCKED PERSONAL SETTINGS
-- free 25K calls per day. Uses separate call for·
-- current weather and for forecast
-- only HC2 location or lat/lon coordinates are used
unlock_appID  = "[redacted]"
unlock_appKey = "[redacted]"

-- WUWEATHER PERSONAL SETTINGS
-- available only for users that connected their PWS with the service
-- must define station ID and personal API key
wdg_stationID = "[redacted]"
wdg_key       = "[redacted]"

```

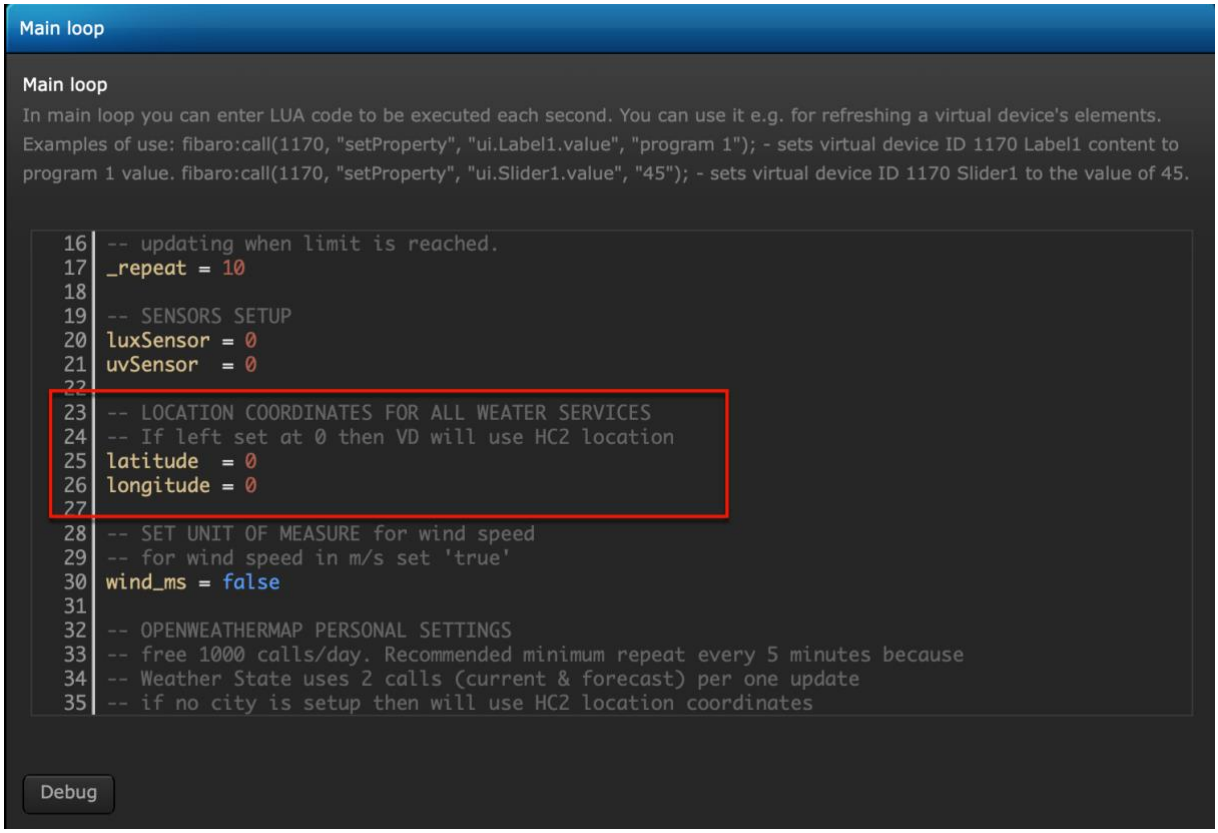
Openweathermap settings are:

- **cityID** – on this link: city.list.json.gz user can download list of all cities ID codes. When user find code for the place just paste ID between quotes.
- **city** – If user prefer to use city name then it goes here e.g. “London”, but name also goes with country code setting
- **country** – here goes country e.g. “uk”
- **opn_key** – here goes API key that user obtained from weather provider

Other services settings are simple and straightforward. All services API key must be provided while city and country are optional.

NOTE - Weather HERE can now be used with API KEY instead of depreciated **appID** and **appCode**.

IMPORTANT NOTE - If no city name is provided then weather module will use either HC2 gateway location or will use user settings for **latitude** and **longitude** in VD main loop:

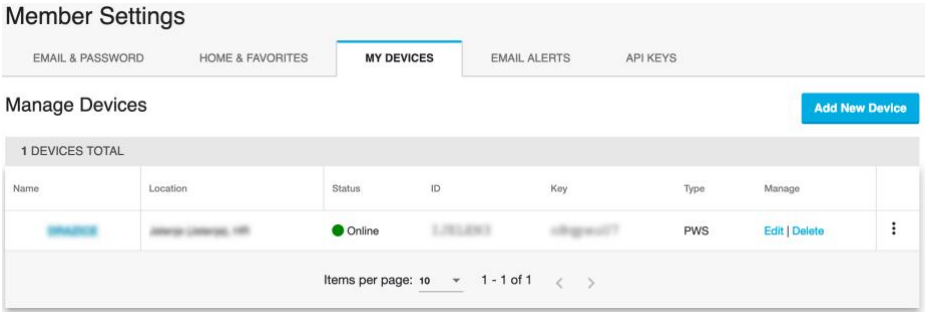


3.9 STEP 9 – WS VD SETTING WUNDERGROUND

Wunderground settings are:

- **wdg_stationID** – here goes personal weather station ID name. This name can be obtained by registering user owned PWS (personal weather station).
- **wdg_key** – here goes API key that can be obtained only by registering PWS.

If user have WeatherFlow Tempest station or any other personal weather station that provides integration with Wunderground then user can register it with Wunderground weather service. First user need to make account on Wunderground and then add weather station and will be provided with the station ID and station Key:



Station will be shown as offline until user provide station ID and Key on his weather station website. Keep obtained station ID available because will be needed for setting up WS VD. After connecting weather station user can obtain API key from Wunderground.

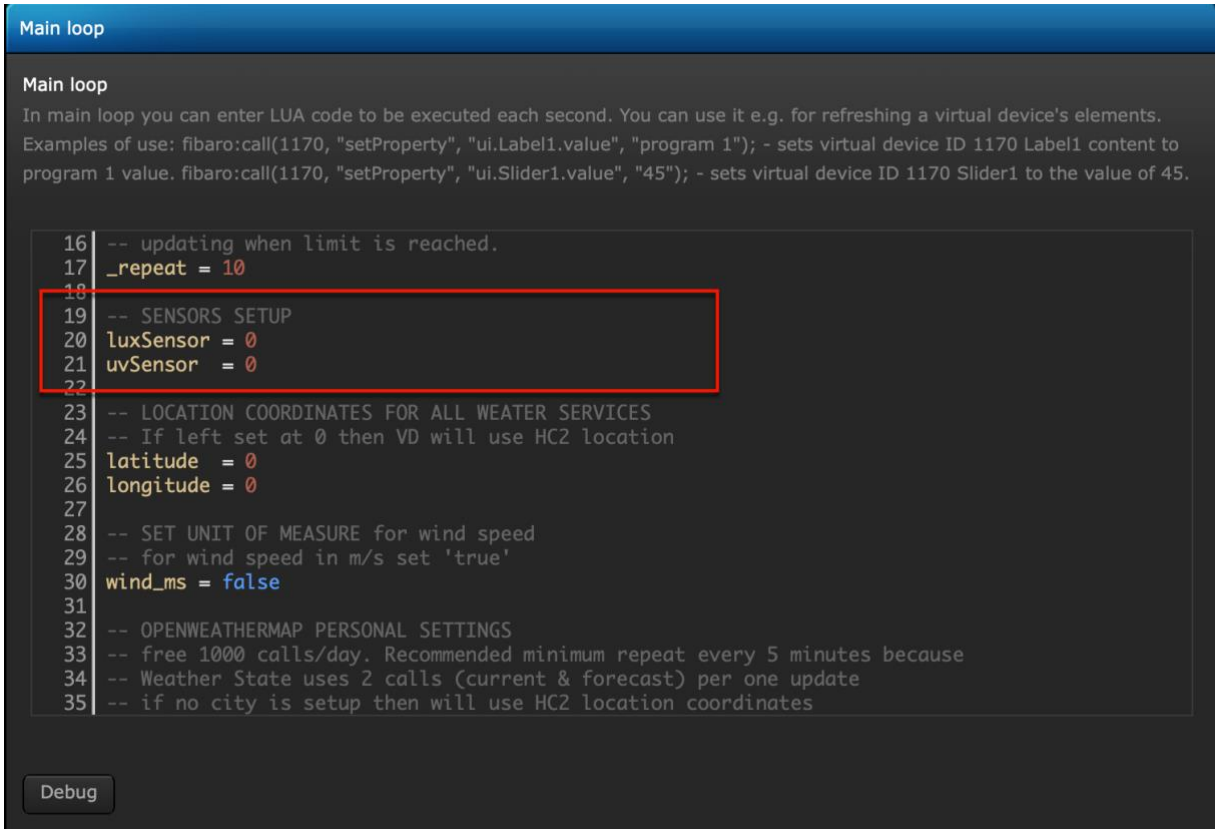
IMPORTANT NOTE – Wunderground weather service can use only users that have registered PWS with their service.

IMPORTANT NOTE – Dark Sky weather service API is not available anymore and therefore removed from this version.

IMPORTANT NOTE – Netatmo partnership with Weather Underground has ended and users are not able to add their Netatmo weather stations to Wunderground anymore.

3.10 STEP 10 – WS VD SETTING ADDITIONAL SENSORS

Please open WS VD for editing and scroll down to the main loop code:

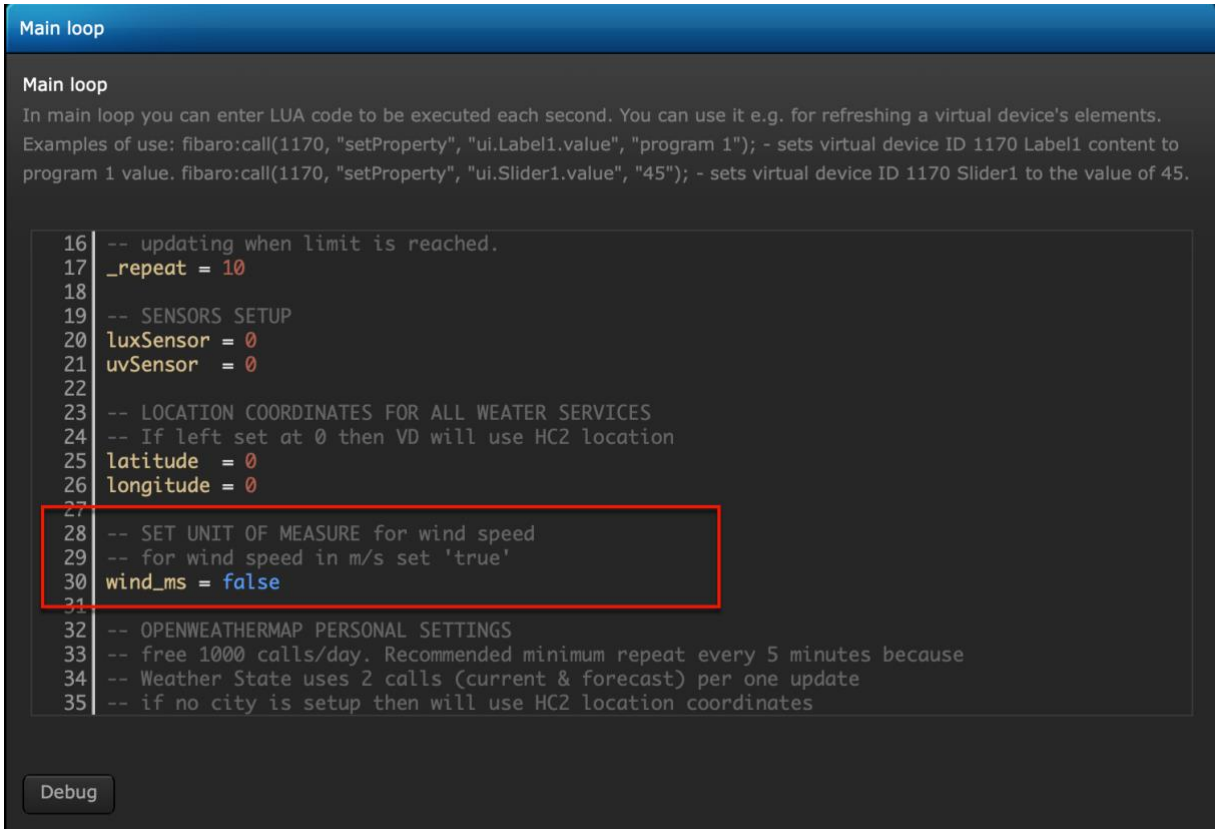


It is possible to setup additional one sensor for measuring light level (lux) and one sensor for measuring UV index. Readings from this sensors will be visible regardless of the weather service selected.

NOTE – If user UV sensor is setup then VD will show user sensor data even selected weather service provide UV index measurement.

3.11 STEP 11 – WS VD SETTING UNITS OF MEASUREMENT

WS VD at every restart checks HC2 location settings temperature unit settings and sets units of measurement accordingly. If this setting is set to “F” Fahrenheit then will use imperial units otherwise metric. User can check in main loop debug window which measurement units are setup after WS VD is restarted.



For wind speed to be calculated in meters per second **wind_ms** must be set to **true**, otherwise leave it as it is. WS VD will show wind speed on GUI and mobile apps, but recalculated to either “km/h” or “mph.

NOTE 1 – If units are not properly read from HC2 location settings then metric system will be used as default. User can then restart WS VD again to get proper settings.

NOTE 2 – HC2 removes decimal places for temperature, humidity and wind speed that are displayed on GUI and mobile apps and readings can slightly differ from what is displayed on WS VD.

IMPORTANT NOTE – Only **WS VD set for station 1** will update GUI weather and any other additional WS VD for other stations will not.

4. WEATHER FORECAST VD INSTALLATION AND SETUP

After importing WS VD it is time to import WF VD. Use same procedure as for WS VD. After importing WF VD user can check main loop debug window and should see following:

```
[DEBUG] 17:33:42: Standalone Weather Forecast VD version 2.8 - (c) 2018-21 Sankotronic
[DEBUG] 17:33:43: [13.04.21]: Global variable WeatherMain_56 created successfully
[DEBUG] 17:33:43: [13.04.21]: Global variable WeatherMain_56 check OK
[DEBUG] 17:33:43: [13.04.21]: Global variable WeatherMain check OK
[DEBUG] 17:33:43: [13.04.21]: VD data added to global variable WeatherMain
[DEBUG] 17:33:43: [13.04.21]: Global variable WeatherForeLang created successfully
[DEBUG] 17:33:43: [13.04.21]: Global variable WeatherForeLang check OK
[DEBUG] 17:33:44: [13.04.21]: Global variable WeatherForeLang updated successfully
[DEBUG] 17:33:44: [13.04.21]: Checking VD icons
[DEBUG] 17:33:44: [13.04.21]: Next message at: 21:00
[DEBUG] 17:33:44: [13.04.21]: VD Idle
```

If user imported more than one WS VD then can also import same number of WF VD and then setup station number same as for WS VD. More details in following chapters.

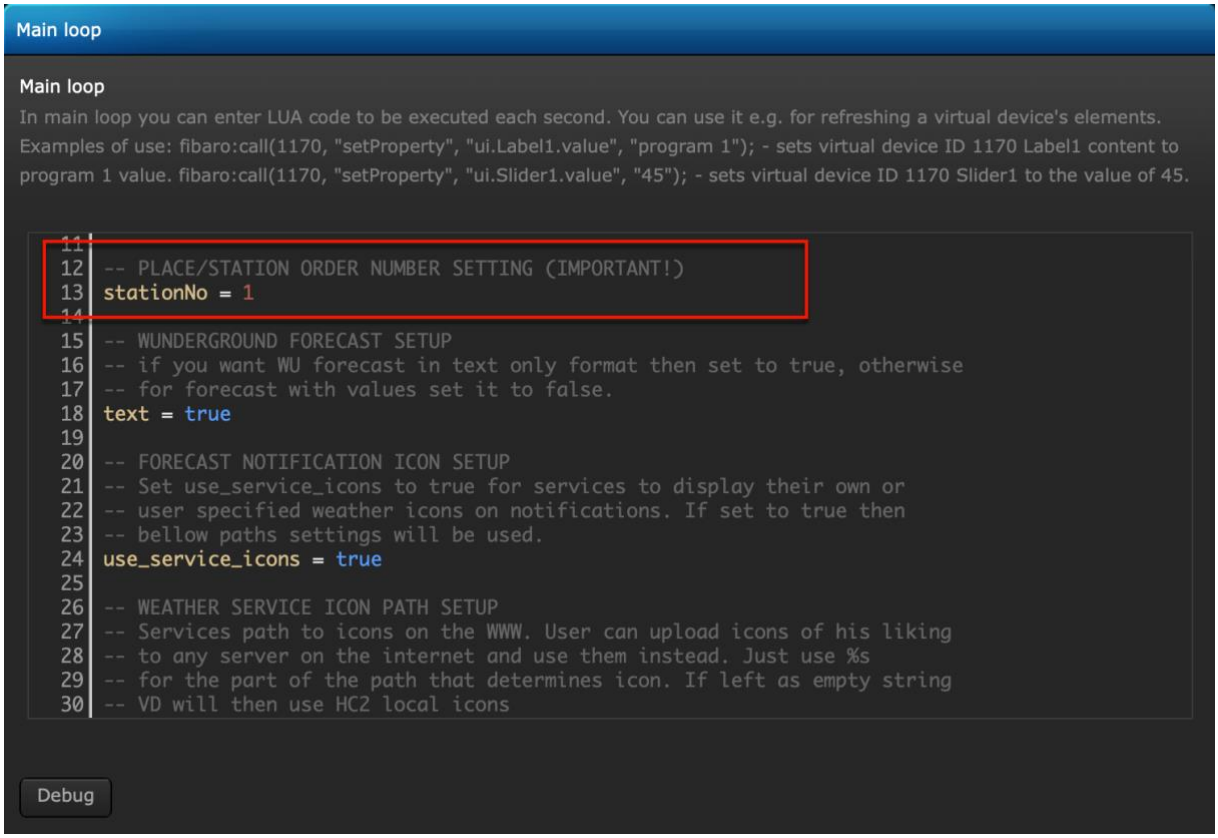
IMPORTANT NOTE - WF VD is updated by WS VD at the repeating time that is setup in WS VD main loop. WF VD will not work without WS VD

After WF VD is installed user can check WS VD update button debug window and will see following debug info when button is pressed:

```
[DEBUG] 17:37:50: [13.04.21]: Found Weather provider plugin ID: 55
[DEBUG] 17:37:50: [13.04.21]: ERROR reading data from weather service
[DEBUG] 17:37:50: [13.04.21]: Updated Forecast VD [56] SA WF VD v2.8
```

4.1 STEP 1 – WF VD SETTING STATION NUMBER

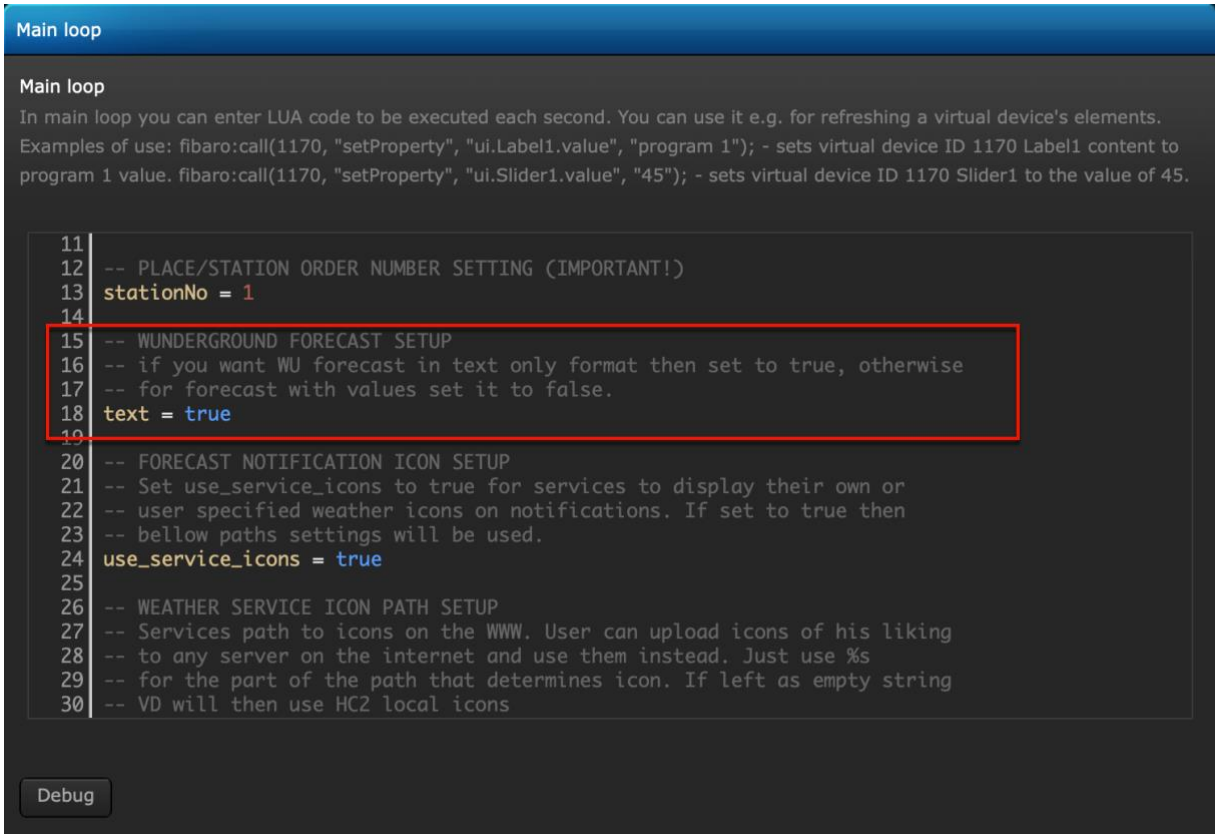
Same as for WS VD it is important to setup station number for each imported WF VD:



Depending on the station setting WF VD will show forecast from the WS VD with same station number.

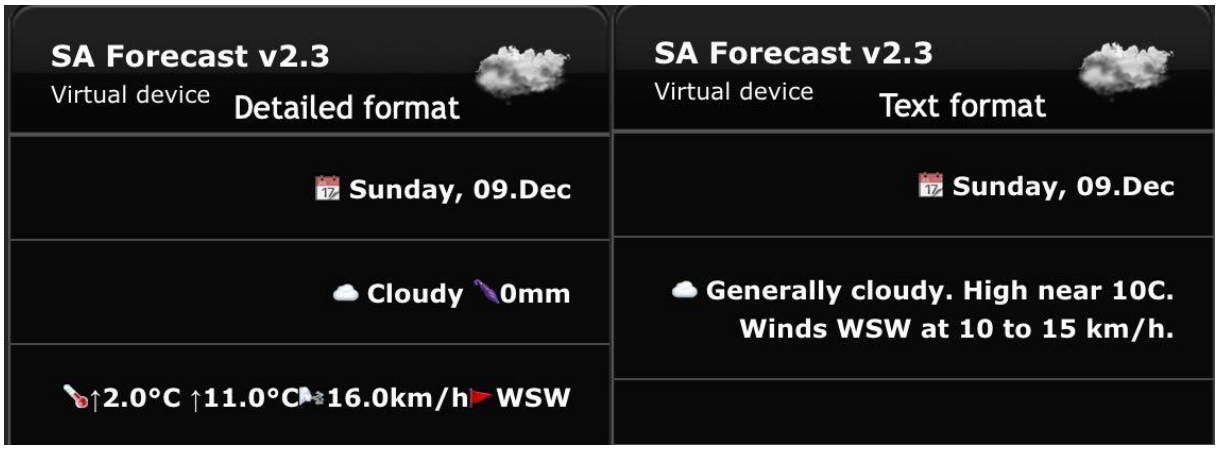
4.2 STEP 2 – WF VD SETTING WUNDERGROUND FORECAST FORMAT

In WF VD window it is possible to set format for the Wunderground forecast to be either as a text or with detailed measurements:



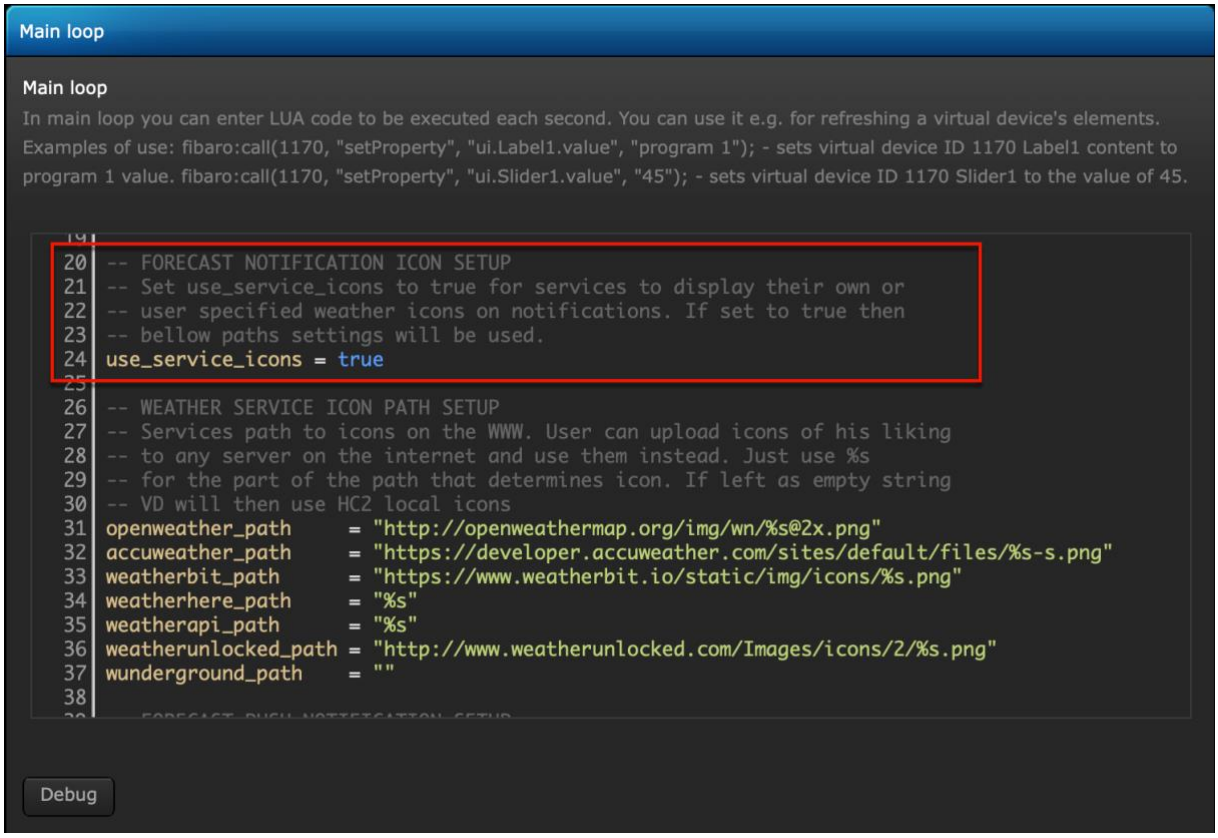
If user prefer to have forecast in text format then set **text** to **true** otherwise for detailed forecast set to **false**.

Example of both formats:



4.3 STEP 3 – WF VD SETTING NOTIFICATION ICONS

With this version of WF VD user can now select which weather icons will be shown on popup notification and can setup path to external icons:



If **use_service_icons** is set to **false** then VD imported weather icons will be shown on popup notification. Since this icons are installed on HC2 they will be visible only when mobile device is connected to HC2 on local network. When user leave house this icons will not show and popup window will show standard icon.

If **use_service_icons** is set to **true** then popup notifications will show icons from either weather service if provided or user icons uploaded to the server on internet. **OpenWeather**, **WeatherBit** and **WeatherHere** services provide their icons and path is already set.

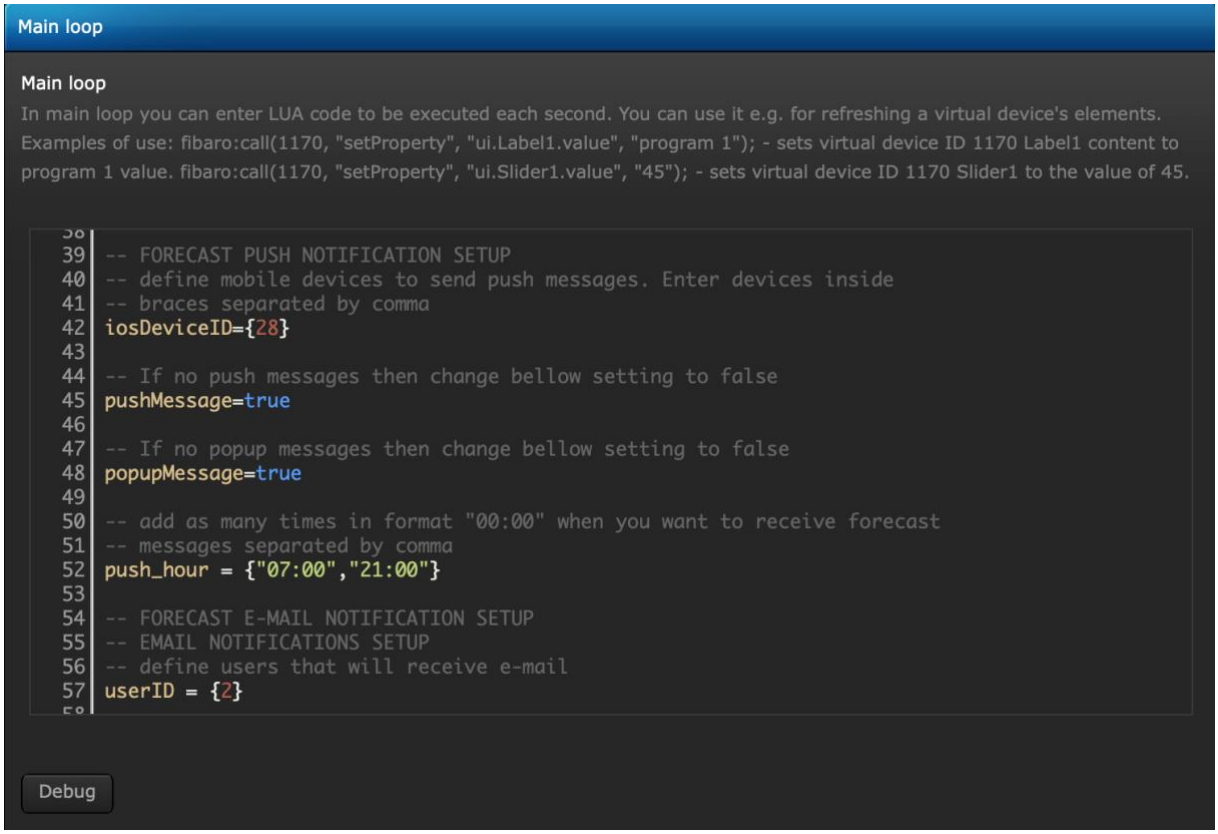
User can setup path to other icons library. In path string use **%s** on the place where icon name is to be inserted. For example it is possible to download icon set from Wunderground weather service here: WU Icons on Google drive and then upload them to user server. After uploading icons user can setup path to this icons as an example:

“http:/<my_server_name>/<icon_folder>/%s.png”

NOTE - If **use_service_icons** is set to **true** but service icon path is left empty **“”** then local VD icons will be used.

4.4 STEP 4 – WF VD SETTING PUSH AND POPUP NOTIFICATIONS SCHEDULE

Open WF VD for editing and scroll main loop code to notification setup part:



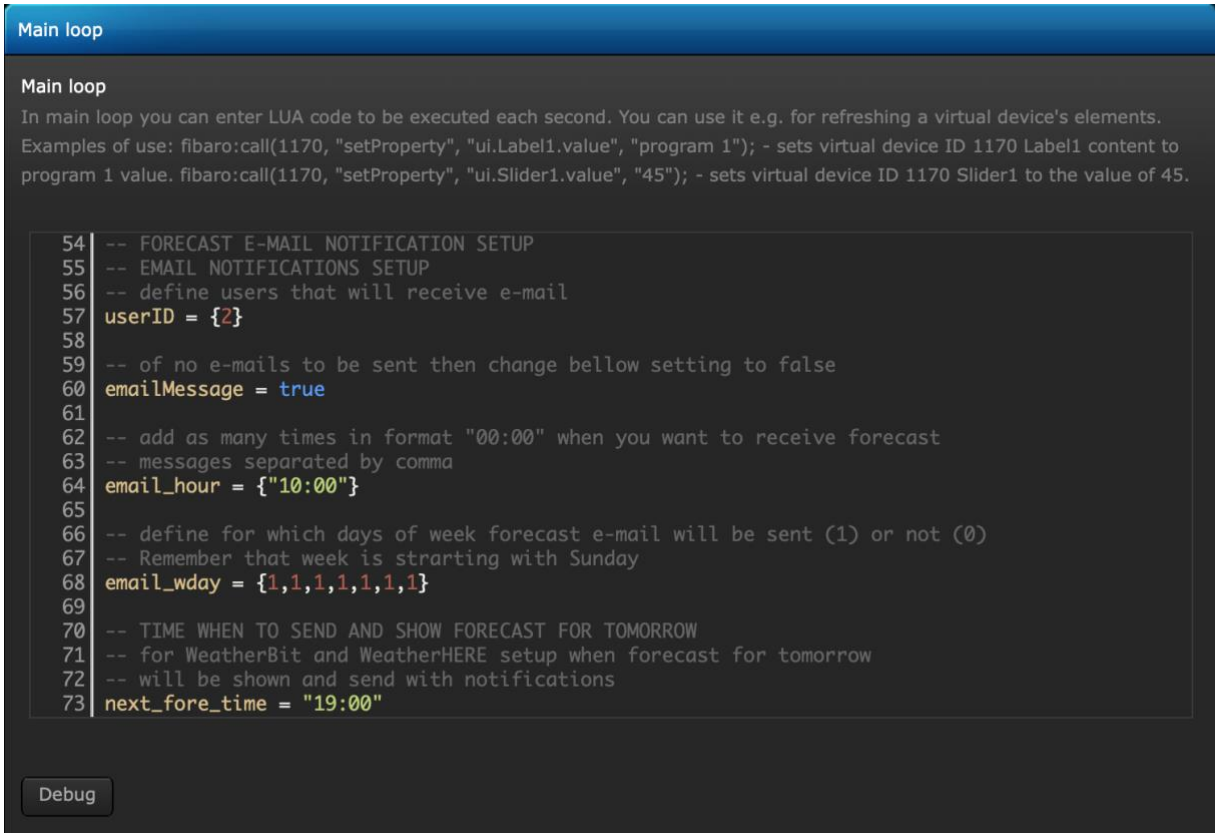
First setup **iosDeviceID** on what mobile devices user like to receive push message by entering mobile devices ID between curled bracket separated by comma. To find out mobile devices ID user can enter this line http://<your_HC2_IP>/docs/#!/iosDevices in browser on computer that is connected to same local network to which HC2 is connected. User need to replace <your_HC2_IP> with HC2 IP address.

User can setup times when WF VD will send forecast notifications using **push_hour** setting. Time to send notifications must be set in format “00:00” inside quotes and separated with comma if there is more than one time set. It is also possible to stop this feature by leaving empty curled brackets {} and then use Main scene FTBE setup or any other code that will press Send notifications button.

User can enable to receive push messages by setting **pushMessage** to true and to receive popup messages by setting **popupMessage** to true, or in any combination.

4.5 STEP 5 – WF VD SETTING E-MAIL NOTIFICATION SCHEDULE

With new version of weather module it is possible to also receive e-mail with complete forecast. User just need to do following setup:



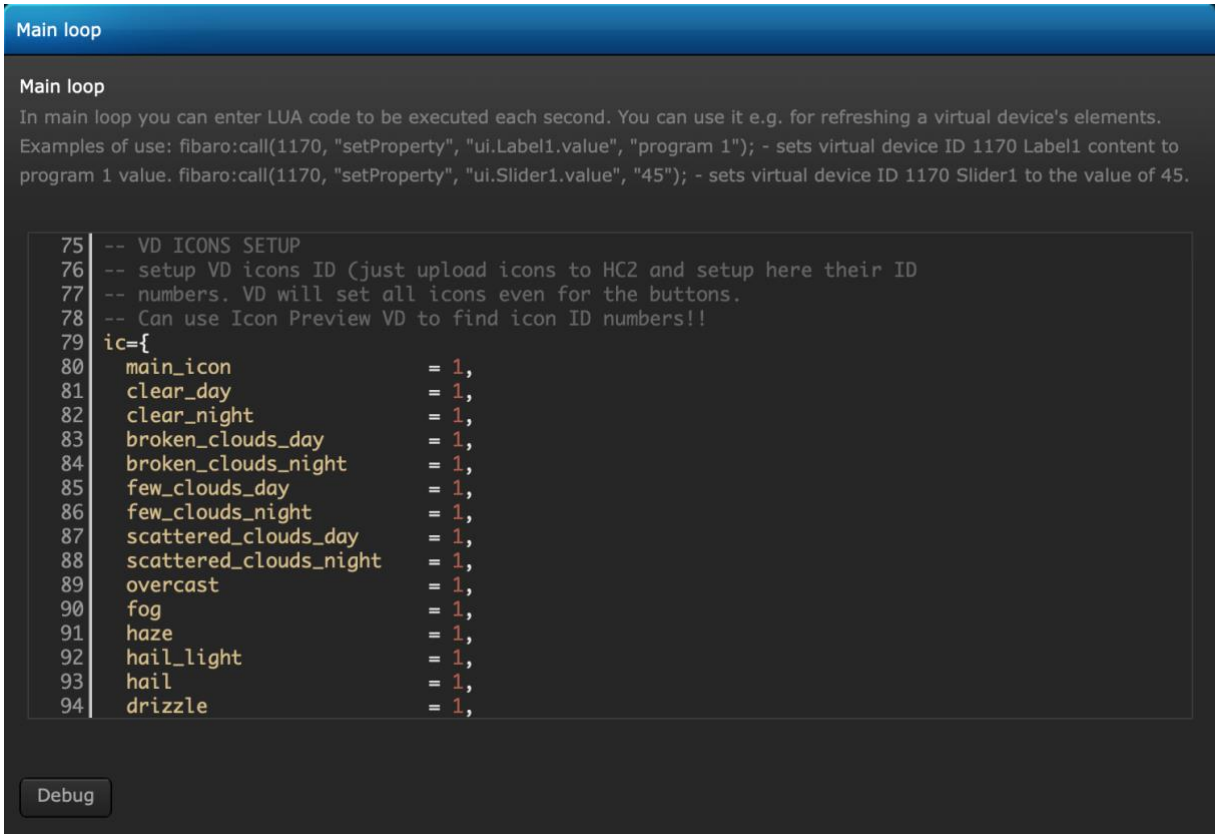
First setup userID for users that will receive forecast e-mail. Just enter user ID between curled brackets separated by comma. To find out HC2 users ID user can enter this line http://<your_HC2_IP>/docs/#!/users in browser on computer that is connected to same local network to which HC2 is connected. User need to replace <your_HC2_IP> with HC2 IP address.

Next step is to setup time at which e-mail will be sent to users **email_hour**. Time to send email must be set in format “00:00” inside quotes and separated with comma if there is more than one time set. It is also possible to stop this feature by leaving empty curled brackets {} and then use Main scene FTBE setup or any other code that will press Send notifications button.

email_wday can be used to setup on which days of the week VD will send email. If 1 is set then will send email and if 0 then no email will be sent. First place in setting is Sunday and then Monday to Saturday.

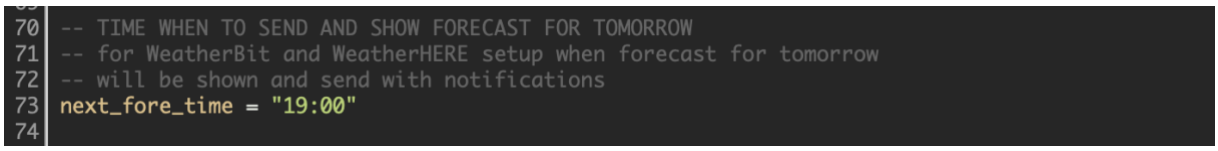
4.6 STEP 6 – WF VD SETTING ICONS

Same as for WS VD user can setup icons for WF VD. Since WF VD uses same set of icons it is recommended just to copy icons settings from WS VD to WF VD by copying this part of code:



4.7 STEP 7 – WF VD SETTING WHEN TO SHOW/NOTIFY NEXT FORECAST

Since some of the weather services does not provide time (hour) when forecast is generated and keep forecast for today until midnight new setting is added to setup at what time today to show forecast for tomorrow:



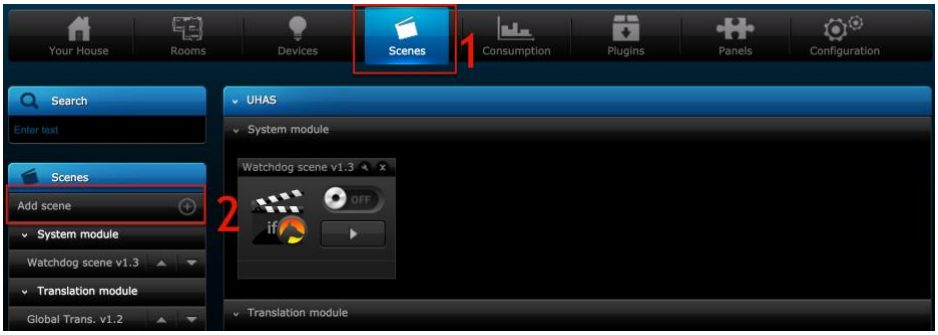
User can change this time to suit his/her needs. For hourly forecasts WF VD checks current time and then sends next relevant forecast e.g. for the next hour or next several hours.

5. WEATHER MODULE SCENE INSTALLATION

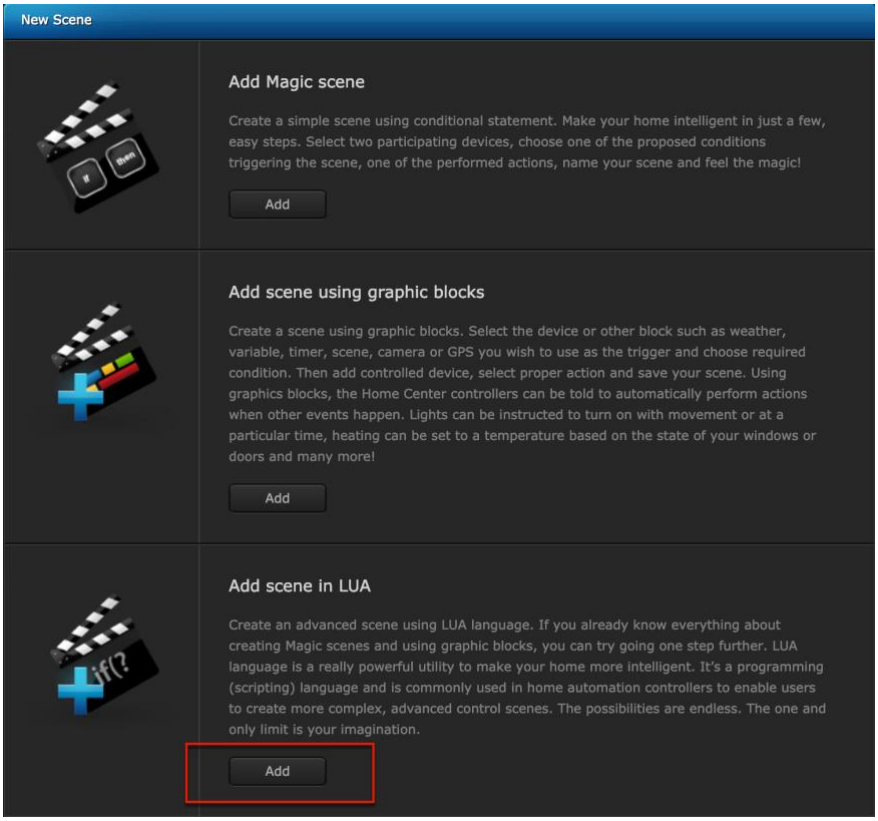
Weather State & Forecast module uses scene to retrieve weather because scene only provides safe HTTPS calls and communication with servers on the internet. If user installed all needed WS VD and WF VD now is time to add scene. Only one scene is needed for any number of installed station pairs of WS VD and WF VD.

5.1 STEP 1 – SCENE INSTALLATION

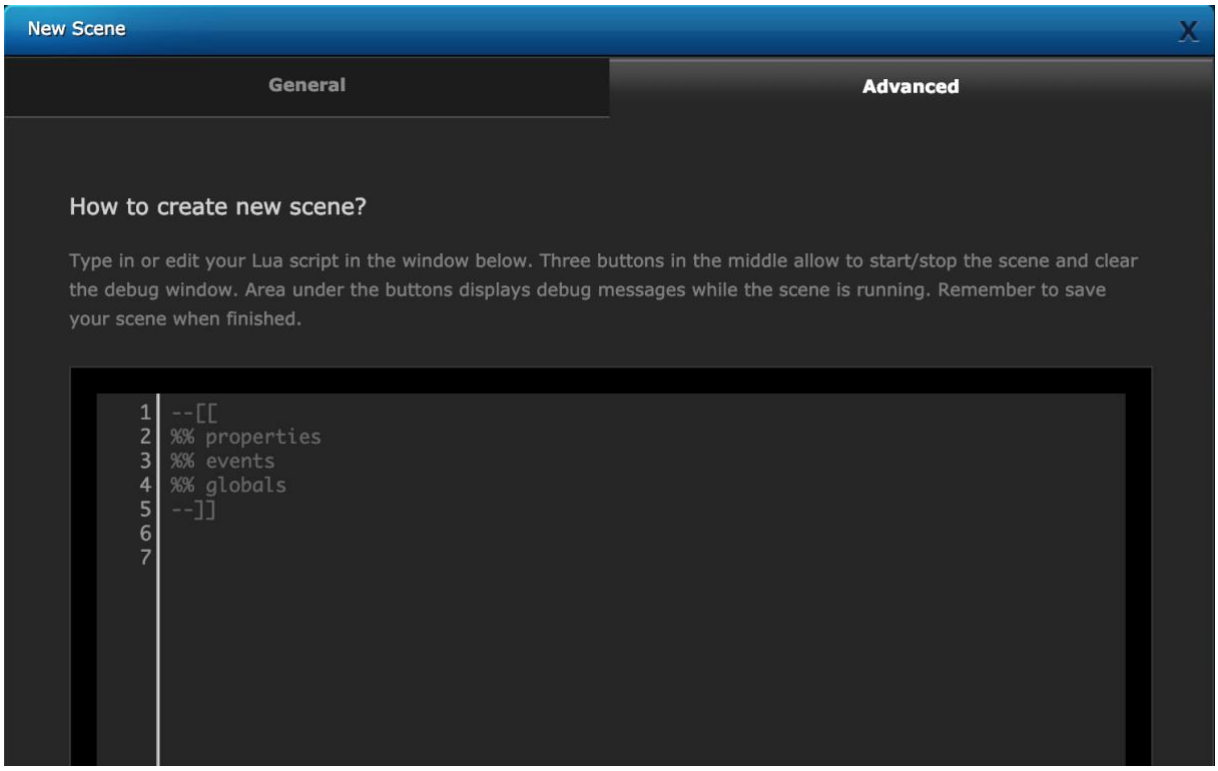
To install scene user first need to select scenes on the HC2 main menu toolbar and then click on Add scene button on left side:



On new window opened user must click on Add button for LUA scenes:

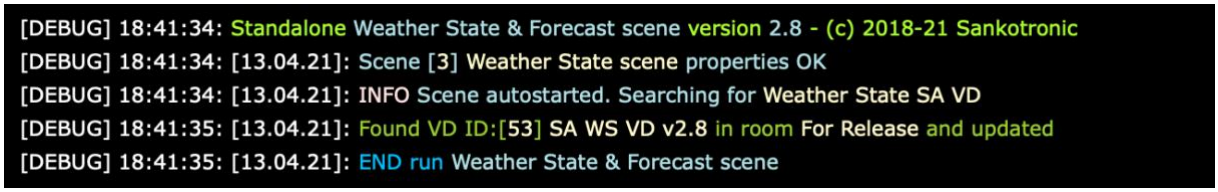


New scene window will open with Advanced tab and code part:



Find scene on computer disk where zip package is expanded and either open file **SA Weather State scene v2.8.lua** with some LUA editor or **SA Weather State scene v2.8.txt** with any plain text editor and copy scene code to new scene window code part.

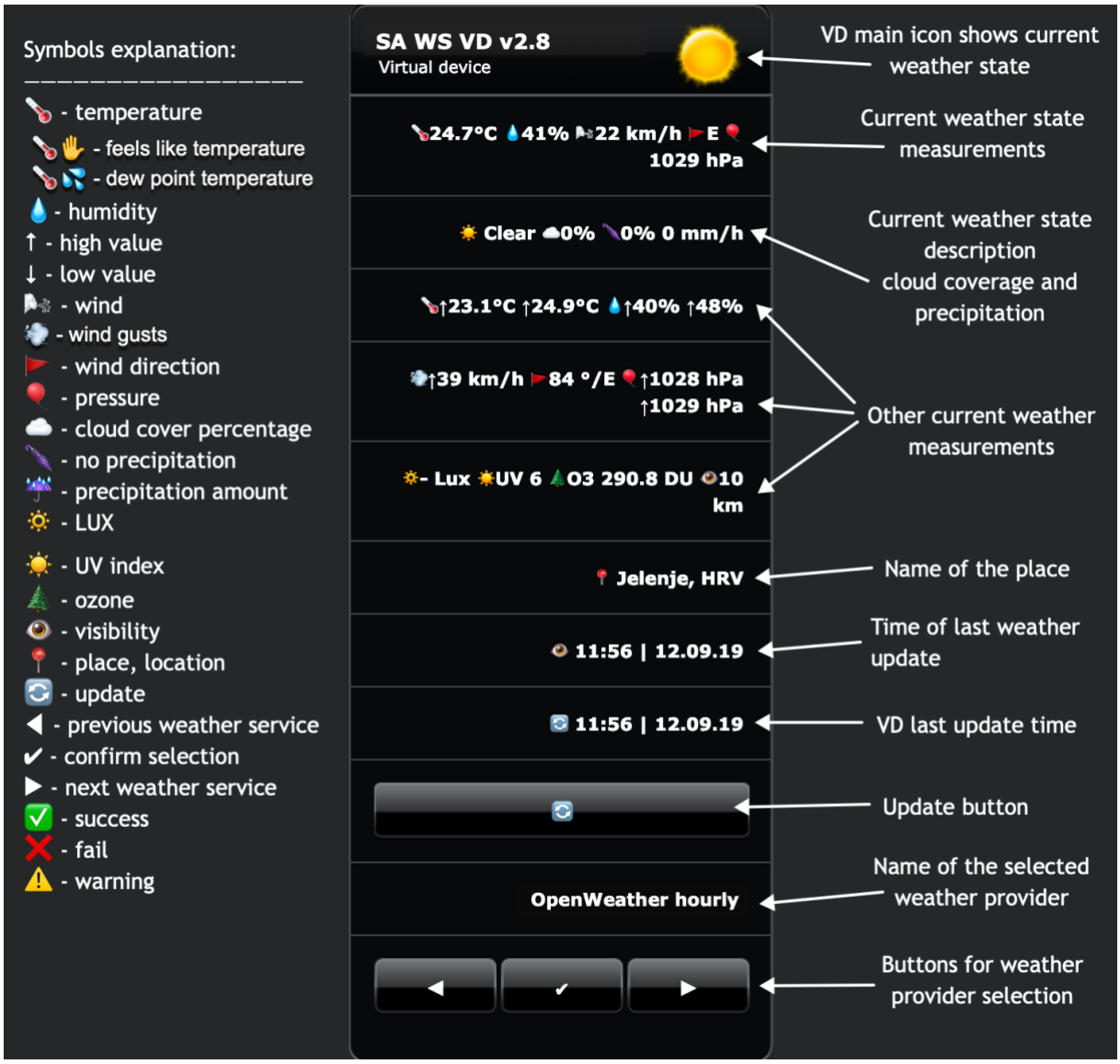
Scene does not have any user settings so user can give scene name, select room and then save it. Scene after saving should run and look for all WS VD installed on HC2 and set scene ID. In scene debug window user can see all found WS VD and that scene ID is set:



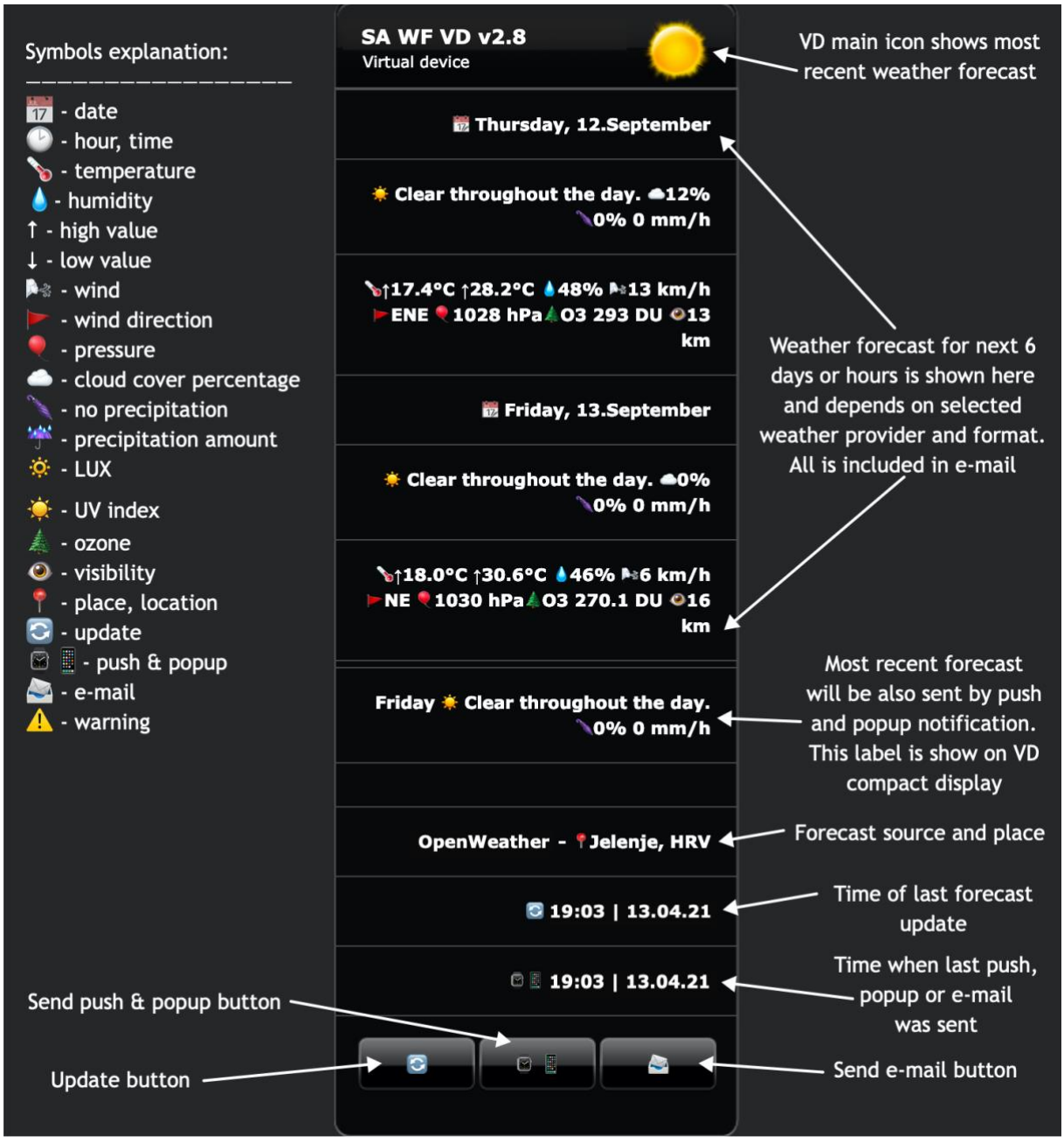
IMPORTANT NOTE – User can now install VDs and scene in any order since VD is now capable of finding scene during first run or when restarted. It is still recommended to first install all needed WS VD and WF VD and then scene last.

6. USING WS VD AND WF VD

And here is short explanation of the WS VD layout:



And here is short explanation of the WF VD layout:



Above layout for WF VD is slightly different depending on user settings. What forecast will be send with notification is decided depending on time when forecast was generated and current time. If current time is greater than forecast time then next forecast is sent. Time when forecast is generated is not visible on VD, but only time for when forecast is valid.

NOTE – Above WF VD picture is truncated and showing only 2 day forecast of 6 in full size.

6.1 – WS VD AND WF VD GLOBAL VARIABLES

All global variables are automatically added when WS VD is imported to HC2. Here is short description:

- ***WeatherMain*** - added automatically to the system by first imported WS VD. It contains only VD ID, original name and role of the all installed weather module VD's for easy search with user code. Role for WS VD is "Weather State SA" and for WF VD is "Weather Fore SA". How to use role is explained in next chapter
- ***WeatherMain_XX*** – XX is replaced by VD ID number. Each imported VD keeps all its settings in their own global variable.
- ***WeatherTable*** – Keeps information about weather type, weather codes and corresponding weather icons.
- ***WeatherState*** – Keeps information about current weather updated from selected weather service
- ***WeatherFore*** – Keeps information about weather forecast updated from selected weather service.
- ***WeatherLang*** – Keeps VD translations for 27 different languages

NOTE - WF VD from version 2.8 is not adding any global variables. All translations are moved to WeatherLang, therefore user can delete WeatherForeLang from the system.

IMPORTANT NOTE

Global variables added by WS and WF VD contain data structured in tables. If **Save** button is used in **Home Center 2 global variable panel** all data in this variables will be destroyed. WS and WF VD checks integrity of global variables and can rebuild them automatically without user intervention and will continue to work properly. Still, it is recommended not to use Save button in HC2 global variable panel. Properly programmed HA system on HC2 does not require user to manually change global variable values in panel using Save button. Such system is UHAS™ (Universal Home Automation System). More information about UHAS™ user can find on Fibaro International forum: https://forum.fibaro.com/topic/25468-sankotronic-lab-universal-home-automation-system-uhas%E2%84%A2/ and soon also on this link: http://www.uhas.eu Also if global variables content is destroyed then after rebuilding default settings will be restored so user will have to redo some of the settings.

6.2 – WS VD AND WF VD USAGE AND CONTROL

WeatherMain global variable contains ID number, original name and role for all WS and WF VD that are imported to HC2. This global can be used to easier find VD in user code and for example to press update button. Here is chunk of code how to search for weather VD for station 1 and to press Update button, send push and send e-mail buttons:

```
23
24  local WSrole = "Weather State SA"
25  local WFrole = "Weather Forecast SA"
26  local stationNo = 1
27  local vdID = "0"
28
29  function findVD(role,station,action)
30    local ok,room=false,""
31    if fibaro:getGlobalValue("WeatherMain")~=nil then
32      local vT=json.decode(fibaro:getGlobalValue("WeatherMain"))
33      for k,v in pairs(vT)do
34        if v.role==role then
35          if fibaro:getGlobalValue("WeatherMain_"..k)~=nil then
36            local vD=json.decode(fibaro:getGlobalValue("WeatherMain_"..k))
37            if vD.sNo==station then vdID=tostring(k)
38              if vD.roomID~=nil then room=fibaro:getRoomName(vD.roomID)else room="Unassigned"end
39              fibaro:debug('Found VD ['..k..'] '..vD.us_name..' in room '..room)
40              if action=="update" then
41                fibaro:call(k,"pressButton",vD.updButton)fibaro:debug("VD updated")
42              elseif action=="push" then
43                fibaro:call(k,"pressButton",vD.msgButton)fibaro:debug("Push & pop notification sent")
44              elseif action="email" then
45                fibaro:call(k,"pressButton",vD.emailButton)fibaro:debug("Push & pop notification sent")
46              else fibaro:debug("Unknown action received. Nothing done")end
47            end
48          end
49        end
50      end
51    else fibaro:debug("Global var 'WeattherMain' not found")end
52    return vdID
53  end
54
55  -- call to update WS VD for station 1
56  vdID=findVD(WSrole,stationNo,"update")
57  if vdID~="0" then
58    -- other user code after VD is updated
59  else fibaro:debug(WSrole.." VD for requested station not found")end
60
61  -- call to update WF VD for station 1
62  vdID=findVD(WFrole,stationNo,"update")
63  if vdID~="0" then
64    -- other user code after VD is updated
65  else fibaro:debug(WFrole.." VD for requested station not found")end
66
67  -- call VD to send push and popup for station 1
68  vdID=findVD(WFrole,stationNo,"push")
69  if vdID~="0" then
70    -- other user code after notifications are sent
71  else fibaro:debug(WFrole.." VD for requested station not found")end
72
73  -- call WF VD to send e-mail for station 1
74  vdID=findVD(WFrole,stationNo,"email")
75  if vdID~="0" then
76    -- other user code after notifications are sent
77  else fibaro:debug(WFrole.." VD for requested station not found")end
78
```

Above code is example how user can write code once and will not have to change it after any weather VD is deleted, moved or added to CH2. Of course there is simpler way to control weather VD's, but keep in mind that this code user will have to change if any of weather VD's is added or deleted:

```
4
5       local wsID = 46
6       local vS=json.decode(fibaro:getGlobalValue("WeatherMain_"..wsID))
7
8       -- update WS VD
9       fibaro:call(wsID,"pressButton",vS.updButton)
10
11      local wfID = 47
12      local vF=json.decode(fibaro:getGlobalValue("WeatherMain_"..wfID))
13
14      -- update WF VD
15      fibaro:call(wfID,"pressButton",vF.updButton)
16      -- WF VD send push and popup notifications
17      fibaro:call(wfID,"pressButton",vF.msgButton)
18      -- WF VD send e-mail notification
19      fibaro:call(wfID,"pressButton",vF.emailButton)
20
```

Global variable ***Weather_main_XX*** where XX is VD ID number contains all other VD settings of which most important for users are VD buttons ID numbers.

6.3 – WEATHER STATE GLOBAL VARIABLE AND HOW TO READ IT

To read current weather state user can use following code:

For OpenWeather current weather data:

```
local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
local stationNo = 1

-- OpenWeather weather data
wT.opn[stationNo].name = "London"
wT.opn[stationNo]._id = "324563"
wT.opn[stationNo].dt = 1568291731
wT.opn[stationNo].fail = false          -- set to true in case of data error valid of all services
wT.opn[stationNo].failch = false        -- can be used as flag for sending error messages
wT.opn[stationNo].w_id = "800"
wT.opn[stationNo].w_desc = "Clear"
wT.opn[stationNo].w_icon = "01d"
wT.opn[stationNo].unit = "metric"
wT.opn[stationNo].temp_unit = "°C"
wT.opn[stationNo].wind_unit = "km/h"
wT.opn[stationNo].distance = "km"
wT.opn[stationNo].press_unit = "hPa"
wT.opn[stationNo].rain_unit = "mm"
wT.opn[stationNo].snow_unit = "cm"
wT.opn[stationNo].temp = 20
wT.opn[stationNo].temp_min = 17
wT.opn[stationNo].temp_max = 23.5
wT.opn[stationNo].humid = 60
wT.opn[stationNo].humid_min = 55
wT.opn[stationNo].humid_max = 72
wT.opn[stationNo].press = 1010
wT.opn[stationNo].press_min = 998
wT.opn[stationNo].press_max = 1012
wT.opn[stationNo].wind = 12
wT.opn[stationNo].wind_max = 18
wT.opn[stationNo].wind_deg = 185
wT.opn[stationNo].wind_dir = "NNE"
wT.opn[stationNo].rain1h = 0
wT.opn[stationNo].rain3h = 0
wT.opn[stationNo].snow1h = 0
wT.opn[stationNo].snow3h = 0
wT.opn[stationNo].visibility = 10
wT.opn[stationNo].clouds = 0
wT.opn[stationNo].lux = 4520
wT.opn[stationNo].uv = 7
```

For AccuWeather current weather data:

```lua

local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
local stationNo = 1

---- AccuWeather data
wT.acc[stationNo].city = ""              -- WS VD will fetch location
wT.acc[stationNo].region = ""            -- WS VD will fetch location
wT.acc[stationNo].country = ""           -- WS VD will fetch location
wT.acc[stationNo].timezone = ""          -- WS VD will fetch location
wT.acc[stationNo].link = ""              -- stores link to the website with current weather data
wT.acc[stationNo].mob_link = ""          -- stores link to the mobile website with current weather data
wT.acc[stationNo].dt = 0                 -- time of the current weather in unix format
wT.acc[stationNo].fail = true            -- set to true in case of data error valid of all services
wT.acc[stationNo].failch = false         -- can be used as flag for sending error messages
wT.acc[stationNo].w_desc = "Sunny"
wT.acc[stationNo].w_icon = ""
wT.acc[stationNo].unit = "metric"
wT.acc[stationNo].temp_unit = "°C"
wT.acc[stationNo].wind_unit = "m/s"
wT.acc[stationNo].distance = "km"
wT.acc[stationNo].height = "m"
wT.acc[stationNo].press_unit = "hPa"
wT.acc[stationNo].rain_unit = "mm/h"
wT.acc[stationNo].snow_unit = "cm/h"
wT.acc[stationNo].temp = 20
wT.acc[stationNo].feel_temp = 21
wT.acc[stationNo].feel_temp_shade = 21
wT.acc[stationNo].dew_temp = 15
wT.acc[stationNo].wind_temp = 15                    -- wind chill temperature
wT.acc[stationNo].bulb_temp = 15                    -- wet bulb temperature
wT.acc[stationNo].temp_min = 17
wT.acc[stationNo].temp_max = 21
wT.acc[stationNo].temp_summary.past6h_min = 21
wT.acc[stationNo].temp_summary.past6h_max = 21
wT.acc[stationNo].temp_summary.past12h_min = 21
wT.acc[stationNo].temp_summary.past12h_max = 21
wT.acc[stationNo].temp_summary.past24h_min = 21
wT.acc[stationNo].temp_summary.past24h_max = 21
wT.acc[stationNo].humid = 40
wT.acc[stationNo].in_humid = 40
wT.acc[stationNo].humid_min = 35
wT.acc[stationNo].humid_max = 60
wT.acc[stationNo].press = 1000
wT.acc[stationNo].press_trend = "Steady"
wT.acc[stationNo].press_min = 980
wT.acc[stationNo].press_max = 1010
wT.acc[stationNo].wind.speed = 5
wT.acc[stationNo].wind.gust = 5
wT.acc[stationNo].wind.deg = 218
wT.acc[stationNo].wind.dir = "NNE"
wT.acc[stationNo].has_precip = true
wT.acc[stationNo].pType = "rain"
wT.acc[stationNo].precip1h = 5
wT.acc[stationNo].precip_summary.past1h = 1
wT.acc[stationNo].precip_summary.past3h = 4
wT.acc[stationNo].precip_summary.past6h = 7
wT.acc[stationNo].precip_summary.past9h = 12
wT.acc[stationNo].precip_summary.past12h = 15
wT.acc[stationNo].precip_summary.past18h = 30
wT.acc[stationNo].precip_summary.past24h = 45
wT.acc[stationNo].visibility = 10
wT.acc[stationNo].obstruction = ""
wT.acc[stationNo].lux = 520
wT.acc[stationNo].uv = 2
wT.acc[stationNo].uv_text = "Low"
wT.acc[stationNo].clouds = 15
wT.acc[stationNo].ceiling = 4500                -- height to clouds

```

For WeatherBit current weather data:

```
local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
local stationNo = 1

-- WeatherBit weather data
wT.wbt[stationNo].city = "city"
wT.wbt[stationNo].country = ""
wT.wbt[stationNo].state = ""
wT.wbt[stationNo].station = ""
wT.wbt[stationNo].timezone = ""
wT.wbt[stationNo].dt = 0
wT.wbt[stationNo].fail = true
wT.wbt[stationNo].failch = false
wT.wbt[stationNo].w_icon = ""
wT.wbt[stationNo].w_code = ""
wT.wbt[stationNo].w_desc = ""
wT.wbt[stationNo].unit = "metric"
wT.wbt[stationNo].temp_unit = "°C"
wT.wbt[stationNo].wind_unit = "km/h"
wT.wbt[stationNo].distance = "km"
wT.wbt[stationNo].press_unit = "hPa"
wT.wbt[stationNo].rain_unit = "mm"
wT.wbt[stationNo].snow_unit = "cm"
wT.wbt[stationNo].solar_unit = "W/m²"
wT.wbt[stationNo].press = 0
wT.wbt[stationNo].slp = 0
wT.wbt[stationNo].press_min = 1300
wT.wbt[stationNo].press_max = 0
wT.wbt[stationNo].wind = 0
wT.wbt[stationNo].wind_max = 0
wT.wbt[stationNo].wind_deg = 0
wT.wbt[stationNo].wind_dir = ""
wT.wbt[stationNo].wind_dir_full = ""
wT.wbt[stationNo].temp = 20
wT.wbt[stationNo].app_temp = 0
wT.wbt[stationNo].temp_min = 60
wT.wbt[stationNo].temp_max = 0
wT.wbt[stationNo].humid = 40
wT.wbt[stationNo].humid_min = 100
wT.wbt[stationNo].humid_max = 0
wT.wbt[stationNo].dewpt = 0
wT.wbt[stationNo].clouds = 0
wT.wbt[stationNo].part_day = ""
wT.wbt[stationNo].visibility = 0
wT.wbt[stationNo].precip = 0
wT.wbt[stationNo].snow = 0
wT.wbt[stationNo].lux = -1
wT.wbt[stationNo].uv = 0
wT.wbt[stationNo].solar_rad = 0
```

For Weather Here current weather data:

```
local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
local stationNo = 1

-- WeatherHere currnet weather data
wT.whr[stationNo].city = "city"
wT.whr[stationNo].country = ""
wT.whr[stationNo].state = ""
wT.whr[stationNo].timezone = ""
wT.whr[stationNo].dt = 0
wT.whr[stationNo].fail = true
wT.whr[stationNo].failch = false
wT.whr[stationNo].icon_name = ""
wT.whr[stationNo].w_code = ""
wT.whr[stationNo].w_desc = ""
wT.whr[stationNo].icon_path = ""
wT.whr[stationNo].unit = "metric"
wT.whr[stationNo].temp_unit = "°C"
wT.whr[stationNo].wind_unit = "km/h"
wT.whr[stationNo].distance = "km"
wT.whr[stationNo].press_unit = "hPa"
wT.whr[stationNo].rain_unit = "cm"
wT.whr[stationNo].snow_unit = "cm"
wT.whr[stationNo].solar_unit = "W/m²"
wT.whr[stationNo].temp = 0
wT.whr[stationNo].temp_desc = ""
wT.whr[stationNo].comfort = 0
wT.whr[stationNo].temp_max = 0
wT.whr[stationNo].temp_min = 60
wT.whr[stationNo].humid = 0
wT.whr[stationNo].humid_min = 100
wT.whr[stationNo].humid_max = 0
wT.whr[stationNo].dewpt = 0
wT.whr[stationNo].precip1h = 0
wT.whr[stationNo].precip3h = 0
wT.whr[stationNo].precip6h = 0
wT.whr[stationNo].precip_desc = ""
wT.whr[stationNo].wind = 0
wT.whr[stationNo].wind_max = 0
wT.whr[stationNo].wind_deg = 0
wT.whr[stationNo].wind_dir = ""
wT.whr[stationNo].wind_desc = ""
wT.whr[stationNo].press = 0
wT.whr[stationNo].press_min = 1300
wT.whr[stationNo].press_max = 0
wT.whr[stationNo].press_trend = ""
wT.whr[stationNo].visibility = 0
wT.whr[stationNo].snowCover = 0
wT.whr[stationNo].uv = -1
wT.whr[stationNo].lux = -1
```

For Wunderground PWS current weather data:

```
local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
local stationNo = 1

-- Wunderground PWS current weather data
wT.wdg[stationNo].city = "city"
wT.wdg[stationNo].country = "HR"
wT.wdg[stationNo]._id = "station"
wT.wdg[stationNo].lat = 0
wT.wdg[stationNo].lon = 0
wT.wdg[stationNo].dt = 0
wT.wdg[stationNo].qcStatus = -1
wT.wdg[stationNo].fail = true
wT.wdg[stationNo].failch = false
wT.wdg[stationNo].w_icon = ""
wT.wdg[stationNo].w_desc = ""
wT.wdg[stationNo].unit = "metric"
wT.wdg[stationNo].temp_unit = "°C"
wT.wdg[stationNo].wind_unit = "km/h"
wT.wdg[stationNo].press_unit = "hPa"
wT.wdg[stationNo].rain_unit = "mm"
wT.wdg[stationNo].snow_unit = "cm"
wT.wdg[stationNo].solar_unit = "W/m²"
wT.wdg[stationNo].temp = 20
wT.wdg[stationNo].temp_min = 60
wT.wdg[stationNo].temp_max = 0
wT.wdg[stationNo].humid = 40
wT.wdg[stationNo].humid_min = 100
wT.wdg[stationNo].humid_max = 0
wT.wdg[stationNo].press = 0
wT.wdg[stationNo].press_min = 1300
wT.wdg[stationNo].press_max = 0
wT.wdg[stationNo].wind = 0
wT.wdg[stationNo].wind_max = 0
wT.wdg[stationNo].wind_deg = 0
wT.wdg[stationNo].wind_dir = ""
wT.wdg[stationNo].gust = 0
wT.wdg[stationNo].gust_max = 0
wT.wdg[stationNo].dewpoint = 0
wT.wdg[stationNo].windChill = 0
wT.wdg[stationNo].heatIndex = 0
wT.wdg[stationNo].visibility = 0
wT.wdg[stationNo].lux = -1
wT.wdg[stationNo].UVindex = -1
wT.wdg[stationNo].solar = ""
wT.wdg[stationNo].precip_rate = 0
wT.wdg[stationNo].precip_total = 0
wT.wdg[stationNo].elevation = 0
```

For Weather API current weather data:

```
294
295 local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
296 local stationNo = 1
297
298 -- Weather API weather data
299 wT.wap[stationNo].city = "city"
300 wT.wap[stationNo].region = ""
301 wT.wap[stationNo].country = ""
302 wT.wap[stationNo].timezone = ""
303 wT.wap[stationNo].dt = 0
304 wT.wap[stationNo].fail = true
305 wT.wap[stationNo].failch = false
306 wT.wap[stationNo].w_desc = ""
307 wT.wap[stationNo].w_icon = ""
308 wT.wap[stationNo].w_code = ""
309 wT.wap[stationNo].unit = "metric"
310 wT.wap[stationNo].temp_unit = "°C"
311 wT.wap[stationNo].wind_unit = "km/h"
312 wT.wap[stationNo].distance = "km"
313 wT.wap[stationNo].press_unit = "hPa"
314 wT.wap[stationNo].rain_unit = "mm"
315 wT.wap[stationNo].snow_unit = "cm"
316 wT.wap[stationNo].temp = 20
317 wT.wap[stationNo].feel_temp = 0
318 wT.wap[stationNo].temp_min = 60
319 wT.wap[stationNo].temp_max = 0
320 wT.wap[stationNo].humid = 40
321 wT.wap[stationNo].humid_min = 100
322 wT.wap[stationNo].humid_max = 0
323 wT.wap[stationNo].press = 0
324 wT.wap[stationNo].press_min = 1300
325 wT.wap[stationNo].press_max = 0
326 wT.wap[stationNo].wind = 0
327 wT.wap[stationNo].wind_max = 0
328 wT.wap[stationNo].wind_gust = 0
329 wT.wap[stationNo].wind_deg = 0
330 wT.wap[stationNo].wind_dir = ""
331 wT.wap[stationNo].precip = 0
332 wT.wap[stationNo].pType = "rain"
333 wT.wap[stationNo].visibility = 0
334 wT.wap[stationNo].lux = -1
335 wT.wap[stationNo].uv = 0
336 wT.wap[stationNo].clouds = 0
337
```

For Weather Unlocked current weather data:

```
340
341 local wT=json.decode(fibaro:getGlobalValue("WeatherState"))
342 local stationNo = 1
343
344 -- Weather Unlocked weather data
345 wT.unl[stationNo].city = "city"
346 wT.unl[stationNo].region = ""
347 wT.unl[stationNo].country = ""
348 wT.unl[stationNo].timezone = ""
349 wT.unl[stationNo].dt = 0
350 wT.unl[stationNo].fail = true
351 wT.unl[stationNo].failch = false
352 wT.unl[stationNo].w_desc = ""
353 wT.unl[stationNo].w_icon = ""
354 wT.unl[stationNo].w_code = ""
355 wT.unl[stationNo].unit = "metric"
356 wT.unl[stationNo].temp_unit = "°C"
357 wT.unl[stationNo].wind_unit = "km/h"
358 wT.unl[stationNo].distance = "km"
359 wT.unl[stationNo].press_unit = "hPa"
360 wT.unl[stationNo].rain_unit = "mm"
361 wT.unl[stationNo].snow_unit = "cm"
362 wT.unl[stationNo].temp = 20
363 wT.unl[stationNo].feel_temp = 0
364 wT.unl[stationNo].temp_min = 60
365 wT.unl[stationNo].temp_max = 0
366 wT.unl[stationNo].humid = 40
367 wT.unl[stationNo].humid_min = 100
368 wT.unl[stationNo].humid_max = 0
369 wT.unl[stationNo].press = 0
370 wT.unl[stationNo].press_min = 1300
371 wT.unl[stationNo].press_max = 0
372 wT.unl[stationNo].wind = 0
373 wT.unl[stationNo].wind_max = 0
374 wT.unl[stationNo].wind_deg = 0
375 wT.unl[stationNo].wind_dir = ""
376 wT.unl[stationNo].precip = 0
377 wT.unl[stationNo].pType = "rain"
378 wT.unl[stationNo].visibility = 0
379 wT.unl[stationNo].lux = -1
380 wT.unl[stationNo].uv = 0
381 wT.unl[stationNo].clouds = 0
382
```

6.4 – WEATHER FORECAST GLOBAL VARIABLE AND HOW TO READ IT

For OpenWeather:

```
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, = 2 for tomorrow

-- OpenWeather weather forecast DAILY data
fT.opn[stationNo].listd[d].dt               -- UNIX time for forecast (numeric)
fT.opn[stationNo].listd[d].wday             -- day of the week name e.g. Monday (string)
fT.opn[stationNo].listd[d].w_id             -- weather forecast code (string)
fT.opn[stationNo].listd[d].w_desc           -- weather forecast text (string)
fT.opn[stationNo].listd[d].w_icon           -- weather forecast icon code (string)
fT.opn[stationNo].listd[d].temp_min         -- forecasted minimum temperature (numeric)
fT.opn[stationNo].listd[d].temp_max         -- forecasted maximum temperature (numeric)
fT.opn[stationNo].listd[d].temp_day         -- feels like temperature for day time (numeric)
fT.opn[stationNo].listd[d].temp_night       -- feels like temperature for night time (numeric)
fT.opn[stationNo].listd[d].temp_eve         -- feels like temperature for evening time (numeric)
fT.opn[stationNo].listd[d].temp_morn        -- feels like temperature for morning time (numeric)
fT.opn[stationNo].listd[d].press            -- forecasted pressure on sea level (numeric)
fT.opn[stationNo].listd[d].humid            -- forecasted relative humidity (numeric)
fT.opn[stationNo].listd[d].wind             -- forecasted wind speed (numeric)
fT.opn[stationNo].listd[d].wind_deg         -- forecasted wind direction, degrees (meteorological) (numeric)
fT.opn[stationNo].listd[d].wind_dir         -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.opn[stationNo].listd[d].clouds           -- clouds coverage in % (numeric)
fT.opn[stationNo].listd[d].pop              -- probability of precipitation (%) (numeric)
fT.opn[stationNo].listd[d].rain             -- precipitation volume of rain (numeric)
fT.opn[stationNo].listd[d].snow             -- precipitation volume of snow (numeric)

-- OpenWeather weather forecast HOURLY data
local h = 1 -- forecast for curent 3 hour period; 2 for next 3 hour period

fT.opn[stationNo].listh[h].dt               -- UNIX time for forecast (numeric)
fT.opn[stationNo].listh[h].wday             -- day of the week name e.g. Monday (string)
fT.opn[stationNo].listh[h].w_id             -- weather forecast code (string)
fT.opn[stationNo].listh[h].w_desc           -- weather forecast text (string)
fT.opn[stationNo].listh[h].w_icon           -- weather forecast icon code (string)
fT.opn[stationNo].listh[h].temp             -- Averige temperature (numeric)
fT.opn[stationNo].listh[h].temp_min         -- forecasted minimum temperature (numeric)
fT.opn[stationNo].listh[h].temp_max         -- forecasted maximum temperature (numeric)
fT.opn[stationNo].listh[h].press            -- forecasted pressure on the ground level (numeric)
fT.opn[stationNo].listh[h].press_sea        -- forecasted pressure on sea level (numeric)
fT.opn[stationNo].listh[h].humid            -- forecasted relative humidity (numeric)
fT.opn[stationNo].listh[h].wind             -- forecasted wind speed (numeric)
fT.opn[stationNo].listh[h].wind_deg         -- forecasted wind direction, degrees (meteorological) (numeric)
fT.opn[stationNo].listh[h].wind_dir         -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.opn[stationNo].listh[h].clouds           -- clouds coverage in % (numeric)
fT.opn[stationNo].listh[h].pop              -- probability of precipitation (%) (numeric)
fT.opn[stationNo].listh[h].rain             -- precipitation volume of rain (numeric)
fT.opn[stationNo].listh[h].snow             -- precipitation volume of snow (numeric)
```

For AccuWeather:

```lua
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, = 2 for tomorrow

-- AccuWeather weather forecast DAILY data
fT.acc[stationNo].listd[d].dt                -- UNIX time for forecast (numeric)
fT.acc[stationNo].listd[d].wday              -- day of the week name e.g. Monday (string)
fT.acc[stationNo].listd[d].w_desc            -- weather forecast text (string)
fT.acc[stationNo].listd[d].w_desc_long       -- weather forecast long text (string)
fT.acc[stationNo].listd[d].w_icon            -- weather forecast icon code (string)
fT.acc[stationNo].listd[d].temp_min          -- forecasted minimum temperature (numeric)
fT.acc[stationNo].listd[d].temp_max          -- forecasted maximum temperature (numeric)
fT.acc[stationNo].listd[d].feel_temp.min     -- Feels like minimum temperature (numeric)
fT.acc[stationNo].listd[d].feel_temp.max     -- Feels like minimum temperature (numeric)
fT.acc[stationNo].listd[d].wind.speed        -- forecasted wind speed (numeric)
fT.acc[stationNo].listd[d].wind.deg          -- forecasted wind direction, degrees (meteorological) (numeric)
fT.acc[stationNo].listd[d].wind.dir          -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.acc[stationNo].listd[d].gust.speed        -- forecasted gust speed (numeric)
fT.acc[stationNo].listd[d].gust.deg          -- forecasted gust direction, degrees (meteorological) (numeric)
fT.acc[stationNo].listd[d].gust.dir          -- forecasted gust direction, adverb e.g. ESE - east-southeast (string)
fT.acc[stationNo].listd[d].clouds            -- clouds coverage in % (numeric)
fT.acc[stationNo].listd[d].chance.precip     -- Percent representing the probability of precipitation (numeric)
fT.acc[stationNo].listd[d].chance.tstorm     -- Percent representing the probability of thunderstorms (numeric)
fT.acc[stationNo].listd[d].chance.rain       -- Percent representing the probability of rain (numeric)
fT.acc[stationNo].listd[d].chance.snow       -- Percent representing the probability of snow (numeric)
fT.acc[stationNo].listd[d].chance.ice        -- Percent representing the probability of ice (numeric)
fT.acc[stationNo].listd[d].amount.total      -- total amount of precipitation (numeric)
fT.acc[stationNo].listd[d].amount.rain       -- total amount of rain (numeric)
fT.acc[stationNo].listd[d].amount.snow       -- total amount of snow (numeric)
fT.acc[stationNo].listd[d].amount.ice        -- total amount of ice (numeric)
fT.acc[stationNo].listd[d].hoursof.precip    -- number of hours of precipitation (numeric)
fT.acc[stationNo].listd[d].hoursof.rain      -- number of hours of rain (numeric)
fT.acc[stationNo].listd[d].hoursof.snow      -- number of hours of snow (numeric)
fT.acc[stationNo].listd[d].hoursof.ice       -- number of hours of ice (numeric)

-- AccuWeather weather forecast HOURLY data
local h = 1 -- forecast for next hour period

fT.acc[stationNo].listh[h].dt                -- UNIX time for forecast (numeric)
fT.acc[stationNo].listh[h].wday              -- day of the week name e.g. Monday (string)
fT.acc[stationNo].listh[h].w_desc            -- weather forecast text (string)
fT.acc[stationNo].listh[h].w_icon            -- weather forecast icon code (string)
fT.acc[stationNo].listh[h].is_day            -- Specifies whether or not it is daylight (true=daylight, false=not daylight) (boolean)
fT.acc[stationNo].listh[h].temp              -- Averige temperature (numeric)
fT.acc[stationNo].listh[h].feel_temp         -- Feels like temperature (numeric)
fT.acc[stationNo].listh[h].dew_temp          -- Dew point temperature (numeric)
fT.acc[stationNo].listh[h].humid             -- forecasted relative humidity (numeric)
fT.acc[stationNo].listh[h].wind.speed        -- forecasted wind speed (numeric)
fT.acc[stationNo].listh[h].wind.deg          -- forecasted wind direction, degrees (meteorological) (numeric)
fT.acc[stationNo].listh[h].wind.dir          -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.acc[stationNo].listh[h].gust              -- forecasted gust speed (numeric)
fT.acc[stationNo].listh[h].visibility        -- Averige visibility (numeric)
fT.acc[stationNo].listh[h].clouds            -- clouds coverage in % (numeric)
fT.acc[stationNo].listh[h].uv                -- Measure of the strength of the ultraviolet radiation from the sun (numeric)
fT.acc[stationNo].listh[h].uv_text           -- Text associated with the uv (string)
fT.acc[stationNo].listh[h].has_precip        -- has true value that indicates the presence of any type of precipitation (boolean)
fT.acc[stationNo].listh[h].chance.precip     -- Percent representing the probability of precipitation (numeric)
fT.acc[stationNo].listh[h].chance.rain       -- Percent representing the probability of rain (numeric)
fT.acc[stationNo].listh[h].chance.snow       -- Percent representing the probability of snow (numeric)
fT.acc[stationNo].listh[h].chance.ice        -- Percent representing the probability of ice (numeric)
fT.acc[stationNo].listh[h].amount.total      -- total amount of precipitation (numeric)
fT.acc[stationNo].listh[h].amount.rain       -- total amount of rain (numeric)
fT.acc[stationNo].listh[h].amount.snow       -- total amount of snow (numeric)
fT.acc[stationNo].listh[h].amount.ice        -- total amount of ice (numeric)
```

For WeatherBIT:

```
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, = 2 for tomorrow

-- WeatherBIT weather forecast DAILY data
fT.wbt[stationNo].list[d].dt                -- UNIX time for forecast (numeric)
fT.wbt[stationNo].list[d].wday              -- day of the week name e.g. Monday (string)
fT.wbt[stationNo].list[d].w_icon            -- weather forecast icon code (string)
fT.wbt[stationNo].list[d].w_code            -- weather forecast code (string)
fT.wbt[stationNo].list[d].w_desc            -- weather forecast text (string)
fT.wbt[stationNo].list[d].wind_gust         -- forecasted gust speed (numeric)
fT.wbt[stationNo].list[d].wind              -- forecasted wind speed (numeric)
fT.wbt[stationNo].list[d].wind_deg          -- forecasted wind direction, degrees (meteorological) (numeric)
fT.wbt[stationNo].list[d].wind_dir          -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.wbt[stationNo].list[d].wind_dir_full     -- forecasted wind direction, full e.g. east-southeast (string)
fT.wbt[stationNo].list[d].temp              -- Average Temperature (numeric)
fT.wbt[stationNo].list[d].temp_min          -- Minimum Temperature (numeric)
fT.wbt[stationNo].list[d].temp_max          -- Maximum Temperature (numeric)
fT.wbt[stationNo].list[d].atemp_min         -- Apparent/"Feels Like" min temperature (numeric)
fT.wbt[stationNo].list[d].atemp_max         -- Apparent/"Feels Like" max temperature (numeric)
fT.wbt[stationNo].list[d].rainP             -- Probability of Precipitation (%) (numeric)
fT.wbt[stationNo].list[d].precip            -- Accumulated liquid equivalent precipitation (numeric)
fT.wbt[stationNo].list[d].snow              -- Accumulated snowfall (numeric)
fT.wbt[stationNo].list[d].snow_depth        -- Snow Depth (numeric)
fT.wbt[stationNo].list[d].slp               -- Average sea level pressure (numeric)
fT.wbt[stationNo].list[d].press             -- Average pressure (numeric)
fT.wbt[stationNo].list[d].dewpt             -- Average dew point temp. (numeric)
fT.wbt[stationNo].list[d].humid             -- Average relative humidity (%) (numeric)
fT.wbt[stationNo].list[d].part_day          -- Part of the day (d = day / n = night) (string)
fT.wbt[stationNo].list[d].clouds            -- Average total cloud coverage (%) (numeric)
fT.wbt[stationNo].list[d].clouds_low        -- Low-level (~0-3km AGL) cloud coverage (%) (numeric)
fT.wbt[stationNo].list[d].clouds_mid        -- Mid-level (~3-5km AGL) cloud coverage (%) (numeric)
fT.wbt[stationNo].list[d].clouds_hi         -- High-level (>5km AGL) cloud coverage (%) (numeric)
fT.wbt[stationNo].list[d].visibility        -- Visibility (default KM) (numeric)
fT.wbt[stationNo].list[d].uv                -- Maximum UV Index (0-11+) (numeric)
```

For WeatherHERE:

```
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, = 2 for tomorrow

-- WeatherHERE weather forecast DAILY data
fT.whr[stationNo].list[d].dt               -- UNIX time for forecast (numeric)
fT.whr[stationNo].list[d].wday             -- day of the week name e.g. Monday (string)
fT.whr[stationNo].list[d].icon_name        -- weather forecast icon name (string)
fT.whr[stationNo].list[d].w_code           -- weather forecast code (string)
fT.whr[stationNo].list[d].w_desc           -- weather forecast text (string)
fT.whr[stationNo].list[d].icon_path        -- weather forecast icon path (string)
fT.whr[stationNo].list[d].wind_desc        -- weather forecast wind text (string)
fT.whr[stationNo].list[d].wind             -- forecasted wind speed (numeric)
fT.whr[stationNo].list[d].wind_deg         -- forecasted wind direction, degrees (meteorological) (numeric)
fT.whr[stationNo].list[d].wind_dir         -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.whr[stationNo].list[d].temp             -- Average Temperature (numeric)
fT.whr[stationNo].list[d].temp_min         -- Minimum Temperature (numeric)
fT.whr[stationNo].list[d].temp_max         -- Maximum Temperature (numeric)
fT.whr[stationNo].list[d].dewpt            -- Average dew point temp. (numeric)
fT.whr[stationNo].list[d].humid            -- Average relative humidity (%) (numeric)
fT.whr[stationNo].list[d].precipP          -- Probability of Precipitation (%) (numeric)
fT.whr[stationNo].list[d].rain_fall        -- Accumulated liquid equivalent precipitation (numeric)
fT.whr[stationNo].list[d].snow_fall        -- Accumulated snowfall (numeric)
fT.whr[stationNo].list[d].press            -- Average pressure (numeric)
fT.whr[stationNo].list[d].part_day         -- Part of the day (D=day or N=night) (string)
fT.whr[stationNo].list[d].sky_desc         -- Description of sky conditions (string)
fT.whr[stationNo].list[d].uv               -- Maximum UV Index (0-10+) (string)
fT.whr[stationNo].list[d].uv_desc          -- UV conditions description (string)
```

For WeatherAPI:

```
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, = 2 for tomorrow

-- WeatherAPI weather forecast DAILY data
fT.wap[stationNo].list[d].dt                -- UNIX time for forecast (numeric)
fT.wap[stationNo].list[d].wday              -- day of the week name e.g. Monday (string)
fT.wap[stationNo].list[d].w_desc            -- weather forecast text (string)
fT.wap[stationNo].list[d].w_icon            -- weather forecast icon path (string)
fT.wap[stationNo].list[d].w_code            -- weather forecast code (string)
fT.wap[stationNo].list[d].temp_min          -- Minimum Temperature (numeric)
fT.wap[stationNo].list[d].temp_max          -- Maximum Temperature (numeric)
fT.wap[stationNo].list[d].temp_avg          -- Average Temperature (numeric)
fT.wap[stationNo].list[d].press             -- Average pressure (numeric)
fT.wap[stationNo].list[d].humid             -- Average relative humidity (%) (numeric)
fT.wap[stationNo].list[d].wind              -- forecasted wind speed (numeric)
fT.wap[stationNo].list[d].wind_max          -- forecasted wind max speed (numeric)
fT.wap[stationNo].list[d].uv                -- Maximum UV Index (numeric)
fT.wap[stationNo].list[d].visibility        -- Averige visibility (numeric)
fT.wap[stationNo].list[d].total_precip      -- Total liquid equivalent precipitation (numeric)
fT.wap[stationNo].list[d].will_rain         -- Will it rain (1 = Yes, 0 = No) (numeric)
fT.wap[stationNo].list[d].will_snow         -- Will it snow (1 = Yes, 0 = No) (numeric)
fT.wap[stationNo].list[d].chance_rain       -- Probability of rain (%) (numeric)
fT.wap[stationNo].list[d].chance_snow       -- Probability of snow (%) (numeric)
```

For Weather Unlocked:

```
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, = 2 for tomorrow

-- Weather Unlocked weather forecast DAILY data
fT.unl[stationNo].listd[d].dt               -- UNIX time for forecast (numeric)
fT.unl[stationNo].listd[d].wday             -- day of the week name e.g. Monday (string)
fT.unl[stationNo].listd[d].w_code           -- weather forecast code (string)
fT.unl[stationNo].listd[d].w_desc           -- weather forecast text (string)
fT.unl[stationNo].listd[d].w_icon           -- weather forecast icon name (string)
fT.unl[stationNo].listd[d].temp_min         -- Minimum Temperature (numeric)
fT.unl[stationNo].listd[d].temp_max         -- Maximum Temperature (numeric)
fT.unl[stationNo].listd[d].press_min        -- Minimum pressure (numeric)
fT.unl[stationNo].listd[d].press_max        -- Maximum pressure (numeric)
fT.unl[stationNo].listd[d].humid_min        -- Minimum relative humidity (numeric)
fT.unl[stationNo].listd[d].humid_max        -- Maximum relative humidity (numeric)
fT.unl[stationNo].listd[d].wind             -- forecasted wind speed (numeric)
fT.unl[stationNo].listd[d].gust             -- forecasted gust speed (numeric)
fT.unl[stationNo].listd[d].clouds           -- Total cloud coverage (%) (numeric)
fT.unl[stationNo].listd[d].pop              -- probability of precipitation percentage (numeric)
fT.unl[stationNo].listd[d].rain             -- precipitation volume of rain (numeric)
fT.unl[stationNo].listd[d].snow             -- precipitation volume of snow (numeric)

-- Weather Unlocked weather forecast HOURLY data
-- h = 1 forecast for first 3 hour period from 01:00 to 03:00
-- h = 3 forecast for 3 hour period from 13:00 to 15:00
-- user need to find value of h for desired 3 hour forecast period
local h = 1

fT.unl[stationNo].listh[h].dt               -- UNIX time for forecast (numeric)
fT.unl[stationNo].listh[h].wday             -- day of the week name e.g. Monday (string)
fT.unl[stationNo].listh[h].w_code           -- weather forecast code (string)
fT.unl[stationNo].listh[h].w_desc           -- weather forecast text (string)
fT.unl[stationNo].listh[h].w_icon           -- weather forecast icon name (string)
fT.unl[stationNo].listh[h].temp             -- Average Temperature (numeric)
fT.unl[stationNo].listh[h].feel_temp        -- Feels like temperature (numeric)
fT.unl[stationNo].listh[h].dew_temp         -- dew point temperature (numeric)
fT.unl[stationNo].listh[h].press            -- Average relative humidity (%) (numeric)
fT.unl[stationNo].listh[h].humid            -- Average relative humidity (%) (numeric)
fT.unl[stationNo].listh[h].wind             -- forecasted wind speed (numeric)
fT.unl[stationNo].listh[h].gust             -- forecasted gust speed (numeric)
fT.unl[stationNo].listh[h].wind_deg         -- forecasted wind direction, degrees (meteorological) (numeric)
fT.unl[stationNo].listh[h].wind_dir         -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.unl[stationNo].listh[h].visibility       -- Averige visibility (numeric)
fT.unl[stationNo].listh[h].clouds           -- Total cloud coverage (%) (numeric)
fT.unl[stationNo].listh[h].chance_precip    -- probability of precipitation percentage (numeric)
fT.unl[stationNo].listh[h].precip           -- precipitation volume (numeric)
fT.unl[stationNo].listh[h].rain             -- precipitation volume of rain (numeric)
fT.unl[stationNo].listh[h].snow             -- precipitation volume of snow (numeric)
fT.unl[stationNo].listh[h].snow_accum       -- amount of fresh snowfall - if accumulated
```

For Wunderground:

```
local fT=json.decode(fibaro:getGlobalValue("WeatherFore"))
local stationNo = 1
local d = 1 -- forecast for today, d = 2 for tonight d = 3 for tomorrow daylight

-- Wunderground weather forecast DAILY data
fT.wdg[stationNo].list[d].wday              -- day of the week name e.g. Monday (string)
fT.wdg[stationNo].list[d].narrative         -- weather forecast very long text (string)
fT.wdg[stationNo].list[d].clouds            -- Total cloud coverage (%) (numeric)
fT.wdg[stationNo].list[d].dayPart           -- Day part name (string)
fT.wdg[stationNo].list[d].icon              -- weather forecast icon code (string)
fT.wdg[stationNo].list[d].text              -- weather forecast long text (string)
fT.wdg[stationNo].list[d].precip_chance     -- probability of precipitation percentage (numeric)
fT.wdg[stationNo].list[d].precip_type       -- type of precipitation e.g. "rain" (string)
fT.wdg[stationNo].list[d].qpf               -- precipitation volume of rain (numeric)
fT.wdg[stationNo].list[d].qpfSnow           -- precipitation volume of snow (numeric)
fT.wdg[stationNo].list[d].snowRng           -- Snow accumulation amount for the 12 hour forecast period
fT.wdg[stationNo].list[d].humid             -- Average relative humidity (%) (numeric)
fT.wdg[stationNo].list[d].temp              -- Average Temperature (numeric)
fT.wdg[stationNo].list[d].temp_hi           -- Heat index Temperature (numeric)
fT.wdg[stationNo].list[d].temp_wc           -- Wind chill Temperature (numeric)
fT.wdg[stationNo].list[d].uv_desc           -- UV index descriptrion (string)
fT.wdg[stationNo].list[d].uv_index          -- Maximum UV index for 12 hour forecast (numeric)
fT.wdg[stationNo].list[d].wind_deg          -- forecasted wind direction, degrees (meteorological) (numeric)
fT.wdg[stationNo].list[d].wind_dir          -- forecasted wind direction, adverb e.g. ESE - east-southeast (string)
fT.wdg[stationNo].list[d].wind_ph           -- The phrase that describes the wind direction and speed (string)
fT.wdg[stationNo].list[d].wind_sp           -- forecasted wind speed (numeric)
fT.wdg[stationNo].list[d].phrase            -- weather forecast short text (string)
```

Thank you for using WS and WF VD and I hope that it will bring many nice moments with your home automation.

Author

7. CHANGELOG AND UPGRADE INSTRUCTIONS

7.1 – VERSION 2.8 CHANGES

- Removed Dark Sky weather service since no longer available
- Added new weather services:
  - AccuWeather
  - Weather API
  - Weather Unlocked
- Weather HERE updated server endpoints so that users can now use new apiKey since appID and appCode are depreciated. Users that have appID and appCode can still continue use them, but it is recommended to get apiKey
- Corrected weather condition on GUI to be properly displayed
- Corrected calculation for temperature, pressure and wind depending on measurement unit setting
- Corrected code that handle error response from servers
- Corrected weather condition icon table to show proper icon for the current or forecasted weather
- Tested and corrected bugs for sending push, popup and e-mail notifications. Since Fibaro still enforce e-mail size limitation it is still possible in some cases that e-mail will not be received due to its length. It all depends on the length of the weather condition text. If this happens please let me know.
- Cleaned more bugs and optimized code.

7.2 – UPGRADE FROM PREVIOUS VERSIONS TO V2.8

1. Before deleting previous version VD please check VD ID number needed to identify global variables. Also save first your VD settings from main loop.
2. Delete all WS VD and WF VD from previous versions
3. Delete global variables ***WeatherMain***, ***WeatherMain_XX*** where XX is the ID number of the WS VD and WF VD virtual devices. Can also delete ***WeatherForeLang*** since it is not required anymore.
4. Install new WS VD and configure
5. Install new WF VD and configure
6. Replace scene code with new code and save
7. Done

7.3 – VERSION V2.8.1 CHANGES

- Weather Unlocked daily and hourly forecast extended to 3 days to better serve Irrigation control. Changes implemented in scene
- Weather API forecast corrected to show rain precipitation on Forecast VD and in notifications.

7.4 - UPGRADE FROM V2.8 TO V2.8.1

- To upgrade from v2.8 to v.2.8.1 please download update package and just paste new code to the scene and Weather Forecast VD buttons with provided code

8. APPENDIX 1

Here is the list of supported languages and their two letter abbreviations to be used in VD main loop language settings:

Languages supported by HC2:

English	=	en	**Polski**	=	pl
Deutch	=	de	**Svenska**	=	sv
Portugues	=	pt	**Italiano**	=	it
Francais	=	fr	**Nederlands**	=	nl
Română	=	ro	**Portugues do Brasil**	=	br
Eesti keel	=	et	**Latvių**	=	lv
中文	=	cn	**Русский**	=	ru
Dansk	=	dk	**Suomalainen**	=	fi
Česky	=	cz	**US English**	=	us
Español	=	es	**Norsk**	=	no

Additional languages supported by this version of WS and WF VD:

български	=	bg	**Hrvatski**	=	hr
Slovenčina	=	sk	**Magyar**	=	hu
Bosanski	=	ba	**Srpski**	=	rs
Slovenščina	=	si		=	

NOTE

Some translations may be incorrect. Users can find all translations in VD main loop and are welcomed to correct them. After correction and saving VD global variable will be automatically updated. Please send me corrections to my e-mail address: zsankovic@gmail.com or private message to Sankotronic on Fibaro International forum: https://forum.fibaro.com

Thank you