```
/**
* Wundeemporground Driver
*
*  Maintained by Derek Osborn
*
*  This driver was originally written by @mattw01 and @Cobra
*  Modified and fixed by @dJOS
*  Additional contributions by @thebearmay @sburke781 @Busthead @swade @kampto
*
*  Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except
*  in compliance with the License. You may obtain a copy of the License at:
*
*      http://www.apache.org/licenses/LICENSE-2.0
*
*  Unless required by applicable law or agreed to in writing, software distributed under the License is distributed
*  on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License
*  for the specific language governing permissions and limitations under the License.
*
*  Last Update 01/16/2025
*
*          v7.2.4 - Only show 3day forecast size on tile when over 1000
*          v7.2.3 - Fix Null Icon on 3day forecast when WU changes from Day to Night Modes
*          v7.2.2 - Added Last Error data to an Attribute so it can be Added to a Tile as an attribute
*          v7.2.1 - Updated Estimated Rain Amounts to Display None instead of 0.0 in when estimate is 0
*          v7.2.0 - Updated Error checking code when network or site issues - Issues Warnings when site and connection issues
*          - and record last error. Only if 5 Warnings in a row will it issue an Error
*          v7.1.7 - Added forecasted rain to 3 Day Forecast Tile
*          v7.1.6 - Added back Feels Like with some improved handling to either wind chill or heat index
*          v7.1.5 - Added rain per day for the last week
*          v7.1.4 - Added Cloud Coverage Forecast of 6 days of AM and PM
*          v7.1.3 - Removed Illuminance capability as it was unusedo either wind chill or heat index
*          v7.1.2 - Removed Station ID from 3 Day Forecast to reduce Hubitats too many characters limitation
*          v7.1.1 - modifed Icon lookup url
*          v7.1.0 - replaced "int" with "java.lang.Integer" to improve compatibility
*          v7.00.5 - changed some attributes from string to number to enable use in RM eg min/max temps
*          v7.00.4 - added Spanish Language support
```

* v7.00.3 - fix odd SolarRadiation value
* v7.00.2 - change several attributes from string to number
* v7.00.1 - Speed Improvements. Improve 3 Day Forecast Tile and removed FeelsLike value since it didn't seem to be correct.
* - Improve Coding Logic. Added ICAO Airport Code for Forecasts.
* - Display an @ sign in forecast day if a WU warning exist for the day. Since WU doesn't provide illuminance, stop using solar radiation as it's value
* v6.10.1 - Changed rain history defaults to enabled with 6/7 days of history
* v6.10.0 - @swade fixed all my LoFi code and made lots of improvements under the hood
* - it is highly recommended that you use the "ClearState" function and then force a manual poll after upgrading to this version.
* - Also, please make sure after you upgrade before you do a poll, you save preferences even if you don't make any changes.
* v6.9.4 - Helps with WU only sending 6 days history for some PWS
* v6.9.3 - additional Logic to deal with only 6 days of rain data by @swade
* v6.9.2 - Enabled the manual entry of Location (lat/long) for Forecasts etc
* v6.8.3 - Added Error Checks when WU doesn't return all days of rain history
* v6.8.2 - Removed option PWS functionality - it just broke too many things - added version reporting
* v6.7.1 - Bug Fixes by @swade
* v6.7.0 - Added Rain History Tile and Today/Tonight forecast header when forecast changes by @swade
* v6.6.0 - Implement Weather Warning Dashboard Tile + made 12:01am default day start for new installs + Forecast Data code restructure
* v6.5.1 - Implement Weather Warnings and Codes
* v6.4.0 - Implement Day/night switching for most forecast items
* v6.3.2 - Fix a rain forecast bug
* v6.3.1 - Bug Fix for null on line 538 error
* v6.3.0 - Added 3 Day Weather Forecast Dashboard tile and additional data ingestion developed by @swade
* v6.2.4 - Not all instances of log.info were checking if txtEnable == true
* v6.2.3 - Replaced logSet with txtEnable to conform with built-in drivers and consolidate Hubitat Preference Manager entries for better user experience
* v6.2.2 - Actually Fixed the Day/Night ForecastDayAfterTomorrow Icon switch over bug
* v6.2.1 - Fixed Day/Night ForecastDayAfterTomorrow Icon switch over bug
* v6.2.0 - Improved formatting, fixed debug.logging and added Station Location to the Tiles
* v6.1.1 - Broke 3 Day FC into individual tiles due to 1024 char limit
* v6.0.2 - Tile bug fixes
* v6.0.1 - Added a 3 Day Forecast Dashboard Tile
* (thanks to @thebearmay for his extensive HTML assistance and @sburke781 for his help with CSS)
* v5.7.0 - Added a 3rd day of forecast data "forecastDayAfterTomorrow" including Icon
* v5.6.5 - Fixed Polling Bug

```
*       v5.6.4 - Removed extra fields due to excess events being generated
*       v5.6.3 - Removed Snow fields due to excess events being generated
*       v5.6.2 - Added 6 new fields - Thunder, Snow & UV
*       v5.6.1 - Minor Bug Fix
*       v5.6.0 - add selectable language eg en-GB or en-US
*       v5.5.0 - WU Icons now hosted on GitHub
*       v5.4.0 - Bug Fixes
*       v5.3.0 - Major to changes forecastToday to auto-switch to night info including fixing icons
to match
*       v5.2.0 - Modified to add forecastToday and forecastTomorrow by Derek Osborn
*       v5.1.0 - Modified to use latitude and longitude from the hub and add cloudCover by
Derek Osborn
*       V5.0.0 - Release by @Cobra
*       V1.0.0 - Original @mattw01 version
*
*/

import groovy.transform.Field
import java.time.LocalTime
def version() {
   return "7.2.4"
}

metadata {
   definition (name: "My Wunderground Driver", namespace: "dJOS", author: "Derek Osborn",
importUrl:
"https://raw.githubusercontent.com/dJOS1475/Hubitat_WU_Driver/main/WU_Driver.groovy") {
       capability "Actuator"
       capability "Sensor"
       capability "Temperature Measurement"
       capability "Illuminance Measurement"
       capability "Relative Humidity Measurement"

       command "poll", [[name:"Start a Manual Poll of Weather Underground Data"]]
            command "clearState", [[name:"Use Only 1 time to Clear State Variables no Longer
Used"]]

       attribute "cloud0day", "string"
       attribute "cloud0AMCoverage", "number"
       attribute "cloud0PMCoverage", "number"
       attribute "cloud1day", "string"
       attribute "cloud1AMCoverage", "number"
       attribute "cloud1PMCoverage", "number"
       attribute "cloud2day", "string"
```

attribute "cloud2AMCoverage", "number"
attribute "cloud2PMCoverage", "number"
attribute "cloud3day", "string"
attribute "cloud3AMCoverage", "number"
attribute "cloud3PMCoverage", "number"
attribute "cloud4day", "string"
attribute "cloud4AMCoverage", "number"
attribute "cloud4PMCoverage", "number"
attribute "cloud5day", "string"
attribute "cloud5AMCoverage", "number"
attribute "cloud5PMCoverage", "number"
attribute "LastErrorDesc", "string"
attribute "LastError", "string"
attribute "forecastPhraseTodayShort", "string"
attribute "dayOrNight", "string"
attribute "formatLanguage", "string"
attribute "formatUnit", "string"
attribute "rainHistoryDays", "number"
attribute "forecastTimeName", "string"
attribute "htmlRainTile", "string"
attribute "precip_Yesterday", "number"
attribute "precip_Last3Days", "number"
attribute "precip_Last5Days", "number"
attribute "precip_Last7Days", "number"
attribute "precip_0", "number"
attribute "precip_0_day", "string"
attribute "precip_1", "number"
attribute "precip_1_day", "string"
attribute "precip_2", "number"
attribute "precip_2_day", "string"
attribute "precip_3", "number"
attribute "precip_3_day", "string"
attribute "precip_4", "number"
attribute "precip_4_day", "string"
attribute "precip_5", "number"
attribute "precip_5_day", "string"
attribute "precip_6", "number"
attribute "precip_6_day", "string"
attribute "temperatureMaxToday", "number"
attribute "temperatureMaxTomorrow", "number"
attribute "temperatureMaxDayAfterTomorrow", "number"
attribute "temperatureMinToday", "number"
attribute "temperatureMinTomorrow", "number"
attribute "temperatureMinDayAfterTomorrow", "number"

```
        attribute "forecastPhraseToday", "string"
        attribute "forecastPhraseTomorrow", "string"
        attribute "forecastPhraseDayAfterTomorrow", "string"
        attribute "precipChanceToday", "number"
        attribute "precipChanceTomorrow", "number"
        attribute "precipChanceDayAfterTomorrow", "number"
        attribute "sunriseTimeLocal", "String"
        attribute "sunsetTimeLocal", "String"
        attribute "html3dayfcst", "string"

        attribute "htmlWarnings", "string"
        attribute "htmlToday", "string"
        attribute "htmlTomorrow", "string"
        attribute "htmlDayAfterTomorrow", "string"
              attribute "today", "string"
              attribute "tomorrow", "string"
        attribute "dayAfterTomorrow", "string"
attribute "uvDescription", "string"
attribute "uvIndex", "number"
attribute "snowRange", "number"
attribute "qpfSnow", "number"
attribute "thunderCategory", "string"
attribute "thunderIndex", "number"
attribute "precipType", "string"
attribute "solarradiation", "number"
attribute "illuminance", "number"
attribute "observation_time", "string"
attribute "weather", "string"
attribute "feelsLike", "number"
attribute "windChill", "number"
attribute "heatIndex", "number"
              attribute "forecastTodayIcon", "string"
attribute "forecastTomorrowIcon", "string"
attribute "forecastDayAfterTomorrowIcon", "string"
              attribute "city", "string"
attribute "state", "string"
attribute "percentPrecip", "number"
attribute "wind_string", "string"
attribute "pressure", "decimal"
attribute "dewpoint", "number"
attribute "visibility", "number"
attribute "forecastHigh", "number"
attribute "forecastLow", "number"
attribute "forecastToday", "string"
```

```
attribute "forecastTomorrow", "string"
attribute "forecastDayAfterTomorrow", "string"
attribute "forecastTemp", "string"
              attribute "forecastShort", "string"
attribute "wind_dir", "string"
              attribute "wind_degree", "string"
attribute "wind_gust", "number"
attribute "precip_rate", "number"
attribute "precip_today", "number"
attribute "wind", "number"
              attribute "windPhrase", "string"
              attribute "windPhraseForecast", "string"
attribute "UV", "number"
   attribute "UVHarm", "string"
attribute "pollsSinceReset", "number"
attribute "temperatureUnit", "string"
attribute "distanceUnit", "string"
attribute "pressureUnit", "string"
attribute "rainUnit", "string"
attribute "summaryFormat", "string"
attribute "alert", "string"
attribute "elevation", "number"
attribute "stationID", "string"
              attribute "stationType", "string"
attribute "weatherSummary", "string"
attribute "weatherSummaryFormat", "string"
attribute "fCstRainToday", "number"
attribute "fCstRainTomorrow", "number"
attribute "fCstRainDayAfterTomorrow", "number"
attribute "moonPhase", "string"
              attribute "humidity", "number"
              attribute "station_location", "string"
attribute "elevation", "number"
attribute "lastUpdateCheck", "string"
attribute "lastPollTime", "string"
attribute "cloudCover", "number"
attribute "weatherWarning", "string"
attribute "weatherWarningCode", "string"
attribute "weatherWarningTomorrow", "string"
attribute "weatherWarningCodeTomorrow", "string"
              attribute "weatherWarningDATomorrow", "string"
attribute "weatherWarningCodeDATomorrow", "string"
attribute "moonIllumination", "number"
attribute "latitude", "decimal"
```

```
                attribute "longitude", "decimal"
        attribute "latitudeCust", "decimal"
                attribute "longitudeCust", "decimal"
                attribute "TempUnit", "string"
                attribute "SpeedUnit", "string"
                attribute "MeasureUnit", "string"

    }
    preferences() {
        section("Query Inputs"){
                        input name: "about", type: "paragraph", element: "paragraph", title:
"Wunderground Driver", description: "v.${version()}"
                        input "apiKey", "text", required: true, title: "API Key"
            input "pollLocation", "text", required: true, title: "Personal Weather Station ID"
            input "pollICAO", "text", required: false, title: "ICAO Airport Code (Forecast)"
                        input "unitFormat", "enum", required: true, title: "Unit Format",  options:
["Imperial", "Metric", "UK Hybrid"]
            if(unitFormat == "UK Hybrid"){input "unitElevation", "bool", required: false, title: "Use
Metric for elevation (m)", defaultValue: false}
            input "language", "enum", required: true, title: "Language",  options: ["US", "GB", "ES"],
defaultValue: US
            input "useIcons", "bool", required: false, title: "Use WU Icons (Optional)", defaultValue:
true
                        if(useIcons){
                        input "iconHeight1", "text", required: true, title: "Icon Height", defaultValue:
100
                        input "iconWidth1", "text", required: true, title: "Icon Width", defaultValue:
100}
            input "autoPoll", "bool", required: false, title: "Enable Auto Poll"
            input "pollInterval", "enum", title: "Auto Poll Interval:", required: false, defaultValue: "5
Minutes", options: ["5 Minutes", "10 Minutes", "15 Minutes", "30 Minutes", "1 Hour", "3 Hours"]
            input "gpsCoords", "bool", title: "Use custom GPS Coordinates for Forecast?",
defaultvalue: false, submitOnChange: true
                        if (gpsCoords) {
                        input "latitudeCust", "text", title: "Enter Latitude in decimals, EX:
37.48644", defaultValue: 0, width: 6, required: false
                        input "longitudeCust", "text", title: "Enter Longitude in decimals, EX:
-121.932309", defaultValue: 0, width: 6, required: false}
            input "weatherwarnings", "bool", title: "Get WU Weather Warnings", required: false,
defaultValue: true
            input "threedayforecast", "bool", title: "Create a 3-Day Forecast Tile", required: false,
defaultValue: true
            input "rainhistory", "bool", title: "Create a 7-Day Rain History Tile", required: false,
defaultValue: true
```

```
        if(threedayforecast && rainhistory){
        input "raindaysdisplay", "enum", title: "Rain History Days to Display On 3-Day Forecast Tile: (Requires 3-day Forecast and Rain History Tiles) (1st/2nd number Days depends on WU returning 6 or 7 day history) ", required: false, defaultValue: "Last 6/7-Days", options: ["None", "Yesterday", "Last 2/3-Days", "Last 4/5-Days", "Last 6/7-Days"]}
                            input "txtEnable", "bool", title: "Enable Detailed logging<br>(auto off in 15 minutes)", required: false, defaultValue: false
                            }
    }
}

@Field static final String ImperialTempUnit='F'
@Field static final String MetricTempUnit='C'
@Field static final String HybridTempUnit='C'
@Field static final String ImperialSpeedUnit='mph'
@Field static final String MetricSpeedUnit='km/h'
@Field static final String ImperialMeasureUnit='in'
@Field static final String MetricMeasureUnit='cm'
@Field static final String HybridMeasureUnit ='cm'

def updated() {
    if(txtEnable){log.debug "updated called"}

    unschedule()
    //reset HTTP Error counter
    //state.HTTPErrorCount = 0

    poll()

    unschedule("changeOver")  //needed to remove unused method

    unschedule("dayRainChange")  //needed to remove unused method

    def pollIntervalCmd = (settings?.pollInterval ?: "5 Minutes").replace(" ", "")
    if(txtEnable == true){log.info "poll IntervalCmd: $pollIntervalCmd"}
    if(autoPoll)
        "runEvery${pollIntervalCmd}"(pollSchedule)

    if(txtEnable){runIn(1800, logsOff)}
}

void clearState() {
    state.clear()
}
```

```
def pollSchedule(){
    poll()
}

def ValueFormating(){
            if(unitFormat == "Imperial"){
        updateTileAttr("formatUnit", "e")
            }
            if(unitFormat == "Metric"){
        updateTileAttr("formatUnit", "m")
            }
            if(unitFormat == "UK Hybrid"){
        updateTileAttr("formatUnit", "h")
            }
            if(language == "US"){
        updateTileAttr("formatLanguage", "en-US")
            }
            if(language == "GB"){
        updateTileAttr("formatLanguage", "en-GB")
            }
            if(language == "ES"){
        updateTileAttr("formatLanguage", "es")
            }
    if(txtEnable == true){log.info "formatUnit = ${device.currentValue('formatUnit')}"}
    if(txtEnable == true){log.info "formatLanguage = ${device.currentValue('formatLanguage')}"}
}

def poll() {
    if(txtEnable == true){log.debug "WU: Poll called"}

    unschedule("changeOver")  //needed to remove unused method
    unschedule("dayRainChange")  //needed to remove unused method

    //remove the illuminance current state as it is not provided by WU
    device.deleteCurrentState('illuminance')
    //clearState()

    locationCoOrds()
    ValueFormating()

    //reset for each check
    state.HTTPErrorFlag = "No"
    state.HTTPErrorTypes = ""
```

```
getObservations()

if (state.HTTPErrorFlag == "No")
{
   GetForecasts()
}

if (state.HTTPErrorFlag == "No")
{
   if(txtEnable == true){log.info "7-Day Rain History: $rainhistory"}
   if (rainhistory)
   {
      GetHistorical()
      rainTile()
   }

   TodayWeatherTile()
   TomorrowWeatherTile()
   DayAfterTomorrowWeatherTile()

   if(txtEnable == true){log.info "3-Day Forecast Tile: $threedayforecast"}
   if(threedayforecast)
   {
      wu3dayfcst()
   }

   if(weatherwarnings)
   {
      WeatherWarningTile()
   }
}

def date = new Date()
if (state.HTTPErrorFlag == "No")
{
   updateTileAttr("lastPollTime", date.format('HH:mm', location.timeZone))
}
else
{
   state.LastHTTPError = date.format('MM/dd/YYYY : HH:mm', location.timeZone)
   state.LastHTTPErrorDesc = state.HTTPError
   // create attributes to show in Tile
   updateTileAttr("LastError", state.LastHTTPError)
```

```
        updateTileAttr("LastErrorDesc", state.LastHTTPErrorDesc)
    }

    //check HTTP error State
    if (state.HTTPErrorFlag == "No")
    {
        state.HTTPError = ""
        state.HTTPErrorTypes = ""
        state.HTTPErrorCount = 0
    }
    else
    {
        state.HTTPErrorCount = state.HTTPErrorCount + 1
    }

    if (state.HTTPErrorCount > 5)
    {
        log.error "(5) Networking or Site Issues in a Row with api.weather.com. Review Warnings
Above."
    }

}

def locationCoOrds(){

    def polllatitude
    def polllongitude

    if(gpsCoords){
        polllatitude = latitudeCust
        polllongitude = longitudeCust
            }
            else{
        polllatitude = location.getLatitude()
        polllongitude = location.getLongitude()
            }

    updateTileAttr("latitude", polllatitude)
    updateTileAttr("longitude", polllongitude)
    if(txtEnable == true){log.info "latitude: $polllatitude"}
    if(txtEnable == true){log.info "longitude: $polllongitude"}
}

def updateTileAttr(String sKey, sValue, String sUnit = ""){
```

```
    sendEvent(name:sKey, value:sValue, unit:sUnit)
    if(txtEnable == true){log.info "String: " + sKey + ": " + sValue + " : " + sUnit}
    //log.info "String: " + sKey + ": " + sValue + " : " + sUnit
}

def updateTileAttr(String sKey, BigDecimal bdValue, String sUnit = ""){
    sValue = String.valueOf(bdValue)
    sendEvent(name:sKey, value:sValue, unit:sUnit)
    if(txtEnable == true){log.info "BigDecimal: " + sKey + ": " + sValue + " : " + sUnit}
    //log.info "BigDecimal: " + sKey + ": " + sValue + " : " + sUnit
}

def getObservations()
{
    //first, go get common oberservations and then return localized data here
    Map localized = getObservationsData()

    if (state.HTTPErrorFlag == "No")
    {
        if(txtEnable == true){log.info "localized-Map: " + localized}

        updateTileAttr("precip_rate", localized.precipRate)
        updateTileAttr("precip_today", localized.precipTotal)
        updateTileAttr("temperature", localized.temp, device.currentValue('TempUnit'))
        updateTileAttr("windChill", localized.windChill, device.currentValue('TempUnit'))

        if(txtEnable == true){log.info "windChill: " + localized.windChill}
        updateTileAttr("heatIndex", localized.heatIndex, device.currentValue('TempUnit'))
        if(txtEnable == true){log.info "heatIndex: " + localized.heatIndex}
        if(localized.heatIndex > localized.temp)
        {
            updateTileAttr("feelsLike", localized.heatIndex, device.currentValue('TempUnit'))
            if(txtEnable == true){log.info "heatIndex (F): " + localized.heatIndex}
        }
        else
        {
            updateTileAttr("feelsLike", localized.windChill, device.currentValue('TempUnit'))
            if(txtEnable == true){log.info "windChill (F): " + localized.windChill}
        }
        updateTileAttr("wind", localized.windSpeed, device.currentValue('SpeedUnit'))
        updateTileAttr("wind_gust", localized.windGust)
        updateTileAttr("dewpoint", localized.dewpt, device.currentValue('TempUnit'))
        updateTileAttr("pressure", localized.pressure)
        updateTileAttr("elevation", localized.elev)
```

```
    }
    else
    {
        state.HTTPErrorTypes = "O"
    }
}

Map getObservationsData()
{
    String wuAPIurl =
"https://api.weather.com/v2/pws/observations/current?format=json&units=${device.currentValue(
'formatUnit')}&stationId=${pollLocation}&apiKey=${apiKey}"
    if(txtEnable == true){log.debug("Getting WU observations from ${wuAPIurl}")}

    Map ret = null
    try {
        httpGet(wuAPIurl) { resp ->
            if (resp?.isSuccess()) {
                try {
                    Map respJSON = resp.getData()
                    if(txtEnable == true){log.info "Observations-Map: " + respJSON.observations[0]}

                    // get top level oberservations
                    def solarRadiation = respJSON.observations.solarRadiation[0]
                    if (solarRadiation ? true : false) // true if not null or zero
                    {
                        if(txtEnable == true){log.debug "solarradiation:
$respJSON.observations.solarRadiation"}
                        updateTileAttr("solarradiation", respJSON.observations.solarRadiation[0])
                    }
                    else
                    {
                        updateTileAttr("solarradiation", "No Data")
                        if(txtEnable == true){log.debug "solarradiation: No Data"}
                    }
                    updateTileAttr("stationID", respJSON.observations.stationID[0])
                    updateTileAttr("stationType", respJSON.observations.softwareType[0])
                    updateTileAttr("station_location", respJSON.observations.neighborhood[0])
                    updateTileAttr("humidity", respJSON.observations.humidity[0])
                    updateTileAttr("observation_time", respJSON.observations.obsTimeLocal[0])
                    updateTileAttr("wind_degree", respJSON.observations.winddir[0])

                    // now return map of localized data
                    if(txtEnable == true){log.info "Resp.Format: $unitFormat"}
```

```
            if (device.currentValue('formatUnit') == 'e'){
                ret = respJSON.observations.imperial[0]
                updateTileAttr("TempUnit", ImperialTempUnit)
                updateTileAttr("SpeedUnit", ImperialSpeedUnit)
                updateTileAttr("MeasureUnit", ImperialMeasureUnit)
            }
            if (device.currentValue('formatUnit') == 'm'){
                ret = respJSON.observations.metric[0]
                updateTileAttr("TempUnit", MetricTempUnit)
                updateTileAttr("SpeedUnit", MetricSpeedUnit)
                updateTileAttr("MeasureUnit", MetricMeasureUnit)
            }
            if (device.currentValue('formatUnit') == 'h'){
                ret = respJSON.observations.uk_hybrid[0]
                updateTileAttr("TempUnit", HybridTempUnit)
                updateTileAttr("SpeedUnit", HybridSpeedUnit)
                updateTileAttr("MeasureUnit", HybridMeasureUnit)
            }
            if(txtEnable == true){log.info "Observations_Localized-Map: " + ret}

        } catch (groovy.json.JsonException ex) {
            state.HTTPErrorFlag = "Yes"
            state.HTTPError = ex.getMessage()
            log.warn("Could not get results from site: ${ex.getMessage()}")
        }
        catch (Exception ex)
        {
            state.HTTPErrorFlag = "Yes"
            state.HTTPError = ex.getMessage()
            log.warn("getObservationsData1 Error: ${ex.getMessage()}")
        }

    } else {
        state.HTTPErrorFlag = "Yes"
        state.HTTPError = "Could not get results from site: ${body}"
        log.warn("Could not get results from site: ${body}")
    }
  }
} catch (java.net.UnknownHostException ex) {
  state.HTTPErrorFlag = "Yes"
  state.HTTPError = ex.getMessage()
  log.warn("Could not connect to the site: ${ex.getMessage()}")
}
```

```
    catch (Exception ex)
    {
        state.HTTPErrorFlag = "Yes"
        state.HTTPError = ex.getMessage()
        log.warn("getObservationsData2 Error: ${ex.getMessage()}")
    }
    return ret
}

def GetForecasts()
{
    Map forecast = GetForecastsData()
    if (state.HTTPErrorFlag == "No")
    {
        if(txtEnable == true){log.info "forecast-Map: " + forecast}
        Forecast(forecast)
    }
    else
    {
        state.HTTPErrorTypes = state.HTTPErrorTypes = "F"
    }
}

Map GetForecastsData()
{
    String wuAPIurl =
"https://api.weather.com/v3/wx/forecast/daily/5day?format=json&units=${device.currentValue('fo
rmatUnit')}&language=${device.currentValue('formatLanguage')}"
    if (gpsCoords ? true : false) // true if not null or zero
    {
            wuAPIurl +=
"&geocode=${device.currentValue('latitude')},${device.currentValue('longitude')}&apiKey=${apiK
ey}"
            if(txtEnable == true){log.info "forecast for: Custom GPS Corrdinates"}
    }
    else
    {
        if (pollICAO ? true : false) // true if not null or zero
        {
            wuAPIurl += "&icaoCode=${pollICAO}&apiKey=${apiKey}"
            if(txtEnable == true){log.info "forecast for: ICAO"}
        }
        else
        {
```

```
        wuAPIurl +=
"&geocode=${device.currentValue('latitude')},${device.currentValue('longitude')}&apiKey=${apiK
ey}"
        if(txtEnable == true){log.info "forecast for: Corrdinates"}
      }
   }

   if(txtEnable == true){log.debug("Getting WU forecast from ${wuAPIurl}")}
   Map ret = null

   try {
      httpGet(wuAPIurl) { resp ->
         if (resp?.isSuccess()) {
            try {
               Map respJSON = resp.getData()
               ret = respJSON
               if(txtEnable == true){log.info "forecasts-Map: " + ret}
            } catch (groovy.json.JsonException ex) {
               state.HTTPErrorFlag = "Yes"
               state.HTTPError = ex.getMessage()
               log.warn("Could not get results from site: ${ex.getMessage()}")
            }
            catch (Exception ex)
            {
               state.HTTPErrorFlag = "Yes"
               state.HTTPError = ex.getMessage()
               log.warn("GetForecastsData1 Error: ${ex.getMessage()}")
            }
         } else {
            state.HTTPErrorFlag = "Yes"
            state.HTTPError = "Could not get results from site: ${body}"
            log.warn("Could not get results from site: ${body}")
         }
      }
   } catch (java.net.UnknownHostException ex) {
      state.HTTPErrorFlag = "Yes"
      state.HTTPError = ex.getMessage()
      log.warn("Could not connect to the site: ${ex.getMessage()}")
   }
   catch (Exception ex)
   {
      state.HTTPErrorFlag = "Yes"
      state.HTTPError = ex.getMessage()
      log.warn("GetForecastsData2 Error: ${ex.getMessage()}")
```

```
    }
    return ret
}

def Forecast(forecast)
{
        //extract the Map data into attributes
    updateTileAttr("temperatureMaxToday", forecast.calendarDayTemperatureMax[0])
    updateTileAttr("temperatureMaxTomorrow", forecast.calendarDayTemperatureMax[1])
    updateTileAttr("temperatureMaxDayAfterTomorrow",
forecast.calendarDayTemperatureMax[2])
    updateTileAttr("temperatureMinToday", forecast.calendarDayTemperatureMin[0])
    updateTileAttr("temperatureMinTomorrow", forecast.calendarDayTemperatureMin[1])
    updateTileAttr("temperatureMinDayAfterTomorrow", forecast.calendarDayTemperatureMin[2])
    updateTileAttr("forecastPhraseTomorrow", forecast.daypart[0].wxPhraseLong[2])
    updateTileAttr("forecastPhraseDayAfterTomorrow", forecast.daypart[0].wxPhraseLong[4])
    updateTileAttr("precipChanceTomorrow", forecast.daypart[0].precipChance[2])
    updateTileAttr("precipChanceDayAfterTomorrow", forecast.daypart[0].precipChance[4])
    updateTileAttr("sunsetTimeLocal", forecast.sunsetTimeLocal[0])
    updateTileAttr("sunriseTimeLocal", forecast.sunriseTimeLocal[0])
    updateTileAttr("today", forecast.dayOfWeek[0])
    updateTileAttr("tomorrow", forecast.dayOfWeek[1])
    updateTileAttr("dayAfterTomorrow", forecast.dayOfWeek[2])
    updateTileAttr("fCstRainTomorrow", forecast.daypart[0].qpf[2])
    updateTileAttr("fCstRainDayAfterTomorrow", forecast.daypart[0].qpf[1])
    updateTileAttr("forecastShort", forecast.narrative[0])
        updateTileAttr("forecastTomorrow", forecast.daypart[0].narrative[2])
        updateTileAttr("forecastDayAfterTomorrow", forecast.daypart[0].narrative[4])
        updateTileAttr("forecastHigh", forecast.temperatureMax[0])
        updateTileAttr("forecastLow", forecast.temperatureMin[0])
        updateTileAttr("moonPhase", forecast.moonPhase[0])

    def cloud6List
    cloud6List = forecast.daypart[0].cloudCover
    if(txtEnable == true){log.info "cloud6List: $cloud6List"}
    //log.info "Cloud Cover: " + cloud6List

    def day6List
    day6List = forecast.dayOfWeek
    if(txtEnable == true){log.info "day6List: $day6List"}
    //log.info "Day CLoud Cover: " + day6List

    CloudCoverDays(day6List, cloud6List)
```

```
    //need WU Day or Night setting so to retrieve correct values
    String DN
    if (forecast.daypart[0].dayOrNight[0] == null)
    {
       DN = "N"
    }
    else
    {
       DN = "D"
    }
          updateTileAttr("dayOrNight", DN)
    if(txtEnable == true){log.info "Day or Night: " + forecast.daypart[0].dayOrNight}
//daypartInitials
    if(txtEnable == true){log.info "D/N Name : " + forecast.daypart[0].daypartName}
//daypartNames
    if(txtEnable == true){log.info "Day or Night: " + DN}

    // Weather Nightime Data
    if(device.currentValue('dayOrNight') == "N"){
       updateTileAttr("forecastTimeName", forecast.daypart[0].daypartName[1])
       updateTileAttr("forecastPhraseToday", forecast.daypart[0].wxPhraseLong[1])
       updateTileAttr("forecastPhraseTodayShort", forecast.daypart[0].wxPhraseShort[1])
          updateTileAttr("precipChanceToday", forecast.daypart[0].precipChance[1])
       updateTileAttr("precipType", forecast.daypart[0].precipType[1])
          updateTileAttr("cloudCover", forecast.daypart[0].cloudCover[1])
                updateTileAttr("uvDescription", forecast.daypart[0].uvDescription[1])
             updateTileAttr("uvIndex", forecast.daypart[0].uvIndex[1])
          updateTileAttr("thunderCategory", forecast.daypart[0].thunderCategory[1])
                updateTileAttr("thunderIndex", forecast.daypart[0].thunderCategory[1])

             updateTileAttr("snowRange", forecast.daypart[0].snowRange[1])
          updateTileAttr("qpfSnow", forecast.daypart[0].qpfSnow[1])
          updateTileAttr("fCstRainToday", forecast.daypart[0].qpf[1])
                updateTileAttr("forecastToday", forecast.daypart[0].narrative[1])
             updateTileAttr("weather", forecast.daypart[0].narrative[1])
          updateTileAttr("wind_dir", forecast.daypart[0].windDirectionCardinal[1])
          updateTileAttr("windPhrase", forecast.daypart[0].windPhrase[1])
                updateTileAttr("windPhraseForecast", forecast.daypart[0].windPhrase[1])

             updateTileAttr("UVHarm", forecast.daypart[0].uvDescription[1])
    }
    else {
       // Weather Daytime Data
       updateTileAttr("forecastTimeName", forecast.daypart[0].daypartName[0])
```

```
                updateTileAttr("forecastPhraseToday", forecast.daypart[0].wxPhraseLong[0])
    updateTileAttr("forecastPhraseTodayShort", forecast.daypart[0].wxPhraseShort[0])
                updateTileAttr("precipChanceToday", forecast.daypart[0].precipChance[0])
        updateTileAttr("precipType", forecast.daypart[0].precipType[0])
      updateTileAttr("cloudCover", forecast.daypart[0].cloudCover[0])
                updateTileAttr("uvDescription", forecast.daypart[0].uvDescription[0])
        updateTileAttr("uvIndex", forecast.daypart[0].uvIndex[0])
      updateTileAttr("thunderCategory", forecast.daypart[0].thunderCategory[0])
                updateTileAttr("thunderIndex", forecast.daypart[0].thunderCategory[0])

        updateTileAttr("snowRange", forecast.daypart[0].snowRange[0])
      updateTileAttr("qpfSnow", forecast.daypart[0].qpfSnow[0])
                updateTileAttr("fCstRainToday", forecast.daypart[0].qpf[0])
        updateTileAttr("forecastToday", forecast.daypart[0].narrative[0])
      updateTileAttr("weather", forecast.daypart[0].narrative[0])
                updateTileAttr("wind_dir", forecast.daypart[0].windDirectionCardinal[0])
        updateTileAttr("windPhrase", forecast.daypart[0].windPhrase[0])
      updateTileAttr("windPhraseForecast", forecast.daypart[0].windPhrase[0])
      updateTileAttr("UVHarm", forecast.daypart[0].uvDescription[0])
  }
  if(weatherwarnings){
    // Weather Warnings Data
    if(device.currentValue('dayOrNight') == "N"){
            String weatherWarning = (forecast.daypart[0].qualifierPhrase[1])
      if(weatherWarning == null){updateTileAttr("weatherWarning", "None")}
            else {updateTileAttr("weatherWarning", weatherWarning)}

                    String weatherWarningCode = (forecast.daypart[0].qualifierCode[1])
      if(weatherWarningCode == null){updateTileAttr("weatherWarningCode", "None")}
                else {updateTileAttr("weatherWarningCode", weatherWarningCode)}
    }
    else{
        String weatherWarning = (forecast.daypart[0].qualifierPhrase[0])
                if(weatherWarning == null){updateTileAttr("weatherWarning", "None")}
                  else {updateTileAttr("weatherWarning", weatherWarning)}


      String weatherWarningCode = (forecast.daypart[0].qualifierCode[0])
                if(weatherWarningCode == null){updateTileAttr("weatherWarningCode", "None")}
                  else {updateTileAttr("weatherWarningCode", weatherWarningCode)}
      }
      String weatherWarningTommorrow = (forecast.daypart[0].qualifierPhrase[2])
                    if(weatherWarningTommorrow ==
null){updateTileAttr("weatherWarningTomorrow", "None")}
```

```
                    else {updateTileAttr("weatherWarningTomorrow",
weatherWarningTommorrow)}

        String weatherWarningCodeTomorrow = (forecast.daypart[0].qualifierCode[2])
                if(weatherWarningCodeTomorrow ==
null){updateTileAttr("weatherWarningCodeTomorrow", "None")}
                        else {updateTileAttr("weatherWarningCodeTomorrow",
weatherWarningCodeTomorrow)}

        String weatherWarningDATomorrow = (forecast.daypart[0].qualifierPhrase[4])
                if(weatherWarningDATomorrow ==
null){updateTileAttr("weatherWarningDATomorrow", "None")}
                        else {updateTileAttr("weatherWarningDATomorrow",
weatherWarningDATomorrow)}

        String weatherWarningCodeDATomorrow = (forecast.daypart[0].qualifierCode[4])
                if(weatherWarningCodeDATomorrow ==
null){updateTileAttr("weatherWarningCodeDATomorrow", "None")}
                        else {updateTileAttr("weatherWarningCodeDATomorrow",
weatherWarningCodeDATomorrow)}
    }
    // Weather Icons Logic
    iconURL1 =
"https://raw.githubusercontent.com/dJOS1475/Hubitat_WU_Driver/main/wuIcons/"

    if(useIcons)
    {
        updateTileAttr("forecastTomorrowIcon", "<img src='" + iconURL1 +
(forecast.daypart[0].iconCode[2]) + ".png" +"' width='" +iconWidth1 +"' height='" +iconHeight1
+"'>")
       updateTileAttr("forecastDayAfterTomorrowIcon", "<img src='" + iconURL1 +
(forecast.daypart[0].iconCode[4]) + ".png" +"' width='" +iconWidth1 +"' height='" +iconHeight1
+"'>")

       if(device.currentValue('dayOrNight') == "N")
       {
          if(forecast.daypart[0].iconCode[1] == null){
             log.warn "WU Null Icon - Night Icon Missing value. Skipped Icon Update (${new
Date()})"
          } else {
                        updateTileAttr("forecastTodayIcon", "<img src='" + iconURL1 +
(forecast.daypart[0].iconCode[1]) + ".png" +"' width='" +iconWidth1 +"' height='" +iconHeight1
+"'>")
             if(txtEnable == true){log.info "Not a Null Night Icon - Normal Icon Used"}
```

```
        }
            }
        else {
        if(forecast.daypart[0].iconCode[0] == null)
        {
            log.warn "WU Null Icon - Day Icon Missing value. Skipped Icon Update (${new
Date()})"
        } else {
                                updateTileAttr("forecastTodayIcon", "<img src='" + iconURL1 +
(forecast.daypart[0].iconCode[0]) + ".png" +"' width='" +iconWidth1 +"' height='" +iconHeight1
+"'>")
            if(txtEnable == true){log.info "Not a Null Day Icon - Normal Icon Used"}
        }
      }
   }
}

def GetHistorical(){
   String wuAPIurl =
"https://api.weather.com/v2/pws/dailysummary/7day?format=json&units=${device.currentValue('f
ormatUnit')}&stationId=${pollLocation}&apiKey=${apiKey}"
   if(txtEnable == true){log.debug("Getting WU historical from ${wuAPIurl}")}
   Map ret = null

   try {
      httpGet(wuAPIurl) { resp ->
         if (resp?.isSuccess()) {
            try {
               Map respJSON = resp.getData()
               if(txtEnable == true){log.info "Historical-Map: " + respJSON.summaries}

               def rain7List
               if(unitFormat == "Imperial")
               {
                  rain7List = (respJSON.summaries.imperial.precipTotal) as String // .toString()
                  if(txtEnable == true){log.info "7DayRain:
$respJSON.summaries.imperial.precipTotal[0]"}
               }
               else
                  if(unitFormat == "Metric")
                  {
                     rain7List = (respJSON.summaries.metric.precipTotal) as String // .toString()
                     if(txtEnable == true){log.info "7DayRain:
$respJSON.summaries.metric.precipTotal[0]"}
```

```
                }
                else
                    if(unitFormat == "UK Hybrid")
                    {
                        rain7List = (respJSON.summaries.uk_hybrid.precipTotal) as String //
.toString()
                        if(txtEnable == true){log.info "7DayRain:
$respJSON.summaries.uk_hybrid.precipTotal[0]"}
                }

                def day7List = []
                day7List = (respJSON.summaries.obsTimeLocal)
                if(txtEnable == true){log.info "day7List: $day7List"}

                CalculatePrecipDays(day7List, rain7List)

            } catch (groovy.json.JsonException ex) {
                state.HTTPErrorFlag = "Yes"
                state.HTTPError = ex.getMessage()
                log.warn("Could not get historic results from site: ${ex.getMessage()}")
            }
            catch (Exception ex)
            {
                state.HTTPErrorFlag = "Yes"
                state.HTTPError = ex.getMessage()
                log.warn("GetHistorical1 Error: ${ex.getMessage()}")
            }

        } else {
            state.HTTPErrorFlag = "Yes"
            state.HTTPError = "Could not get results from site: ${body}"
            log.warn("Could not get historic results from site: ${body}")
        }
    }
  } catch (java.net.UnknownHostException ex) {
    state.HTTPErrorFlag = "Yes"
    state.HTTPError = ex.getMessage()
    log.warn("Could not connect to the site: ${ex.getMessage()}")
  }
  catch (Exception ex)
  {
    state.HTTPErrorFlag = "Yes"
    state.HTTPError = ex.getMessage()
    log.warn("GetHistorical2 Error: ${ex.getMessage()}")
```

```
    }

    if (state.HTTPErrorFlag == "Yes")
    {
        state.HTTPErrorTypes = state.HTTPErrorTypes = "H"
    }

}

def CalculatePrecipDays(day7List, rain7List) {
    if(txtEnable == true){log.info "day7List: $day7List"}

    java.lang.Integer daysCount = day7List.size()
    updateTileAttr("rainHistoryDays", daysCount)
    if(txtEnable == true){log.info "Count: $daysCount"}

    if(txtEnable == true){log.info "Orig: $rain7List"}
    rain7List = rain7List.replace("[","")
    rain7List = rain7List.replace("]","")
    if(txtEnable == true){log.info "Repl: $rain7List"}

    //get precip days names
    PrecipDaysofWeek()

    try{
        def BigDecimal bd6

        if (daysCount == 7)
        {
            (day6, day5, day4, day3, day2, day1, day0) = rain7List.tokenize( ',' )
            if(txtEnable == true){log.info "Day 6: $day6, Day 5: $day5, Day 4: $day4, Day 3: $day3,
Day 2: $day2, Day 1: $day1, Day 0: $day0"}

            bd6 = GetAmountRain(day6)
        }

        if (daysCount == 6)
        {
            (day5, day4, day3, day2, day1, day0) = rain7List.tokenize( ',' )
            if(txtEnable == true){log.info "Day 5: $day5, Day 4: $day4, Day 3: $day3, Day 2: $day2,
Day 1: $day1, Day 0: $day0"}

            bd6 = 0.00
        }
```

```
updateTileAttr("precip_6", bd6)

def BigDecimal bd5
bd5 = GetAmountRain(day5)
updateTileAttr("precip_5", bd5)

def BigDecimal bd4
bd4 = GetAmountRain(day4)
updateTileAttr("precip_4", bd4)

def BigDecimal bd3
bd3 = GetAmountRain(day3)
updateTileAttr("precip_3", bd3)

def BigDecimal bd2
bd2 = GetAmountRain(day2)
updateTileAttr("precip_2", bd2)

def BigDecimal bd1
bd1 = GetAmountRain(day1)
updateTileAttr("precip_1", bd1)

def BigDecimal bd0
bd0 = GetAmountRain(day0)
updateTileAttr("precip_0", bd0)

if(txtEnable == true){log.info "$bd6, $bd5, $bd4, $bd3, $bd2, $bd1, $bd0"}
BigDecimal bdAll7 = bd6 + bd5 + bd4 + bd3 + bd2 + bd1 + bd0
String bdString7 = String.valueOf(bdAll7)
if(txtEnable == true){log.info "7Days: " + bdString7}
updateTileAttr("precip_Last7Days", bdString7)

BigDecimal bdAll5 = bd5 + bd4 + bd3 + bd2 + bd1 + bd0
String bdString5 = String.valueOf(bdAll5)
if(txtEnable == true){log.info "5Days: " + bdString5}
updateTileAttr("precip_Last5Days", bdString5)

BigDecimal bdAll3 = bd3 + bd2 + bd1 + bd0
String bdString3 = String.valueOf(bdAll3)
if(txtEnable == true){log.info "3Days: " + bdString3}
updateTileAttr("precip_Last3Days", bdString3)

BigDecimal bdYesterday = bd1
```

```
        String bdStringYesterday = String.valueOf(bdYesterday)
        if(txtEnable == true){log.info "Yesterday: " + bdStringYesterday}
        updateTileAttr("precip_Yesterday", bdStringYesterday)
    }
    catch (Exception e)
    {
        log.warn "WU did not return all rain values on this attempt. Missing at least 1 day's rain.
Will retry on next interval."
        log.warn "error: $e"
    }
}

def CloudCoverDays(day6List, cloud6List) {
    if(txtEnable == true){log.info "day6List: $day6List"}
    if(txtEnable == true){log.info "cloud6List: $cloud6List"}
    //log.info "day6List: $day6List"
    //log.info "cloud6List: $cloud6List"

    def x = 0
    def z = 0
    if(cloud6List[0] == null)
    {
        x = 1
    }

    day6List.each { day ->
        if(x == 1)
        {
            updateTileAttr("cloud${z}day", day)
            //log.info "cloud${z}day = " + day
            if(txtEnable == true){log.info "cloud${z}day = " + day}
            updateTileAttr("cloud${z}AMCoverage", " ")
            //log.info "cloud${z}AMCoverage = " + " "
            if(txtEnable == true){log.info "cloud${z}AMCoverage = " + " "}
                updateTileAttr("cloud${z}PMCoverage", cloud6List[x])
            //log.info "cloud${z}PMCoverage = " + cloud6List[x]
            if(txtEnable == true){log.info "cloud${z}PMCoverage = " + cloud6List[x]}
            x += 1
        } else {
            updateTileAttr("cloud${z}day", day)
            //log.info "cloud${z}day = " + day
            if(txtEnable == true){log.info "cloud${z}day = " + day}
                updateTileAttr("cloud${z}AMCoverage", cloud6List[x])
            //log.info "cloud${z}AMCoverage = " + cloud6List[x]
```

```
            if(txtEnable == true){log.info "cloud${z}AMCoverage = " + cloud6List[x]}
            updateTileAttr("cloud${z}PMCoverage", cloud6List[x+1])
            //log.info "cloud${z}PMCoverage = " + cloud6List[x+1]
            if(txtEnable == true){log.info "cloud${z}PMCoverage = " + cloud6List[x+1]}
            x += 2
        }
        z +=1
    }
}

BigDecimal GetAmountRain(sDay)
{
    BigDecimal bdRain = 0.00
    if (sDay.trim() != 'null' && sDay.trim() != '')
    {
        if (sDay != null) {bdRain = sDay.toBigDecimal()} else {bdRain = 0}
    }
    return bdRain
}

String PrecipDaysofWeek()
{
    //get precip days names
    String[] precipDays = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
"Saturday"]

    def mydate = new Date()
    def todayIndex = mydate[Calendar.DAY_OF_WEEK]
    if(txtEnable == true){log.info "Today Index: $todayIndex"}
    def myday = precipDays[todayIndex - 1]
    if(txtEnable == true){log.info "Today is: $myday"}

    if (todayIndex == 1) //Sunday
    {
        updateTileAttr("precip_0_day", "Sunday")      //today
        updateTileAttr("precip_1_day", "Saturday")    //yesteray
        updateTileAttr("precip_2_day", "Friday")
        updateTileAttr("precip_3_day", "Thursday")
        updateTileAttr("precip_4_day", "Wednesday")
        updateTileAttr("precip_5_day", "Tuesday")
        updateTileAttr("precip_6_day", "Monday")
    }
    if (todayIndex == 2) //Monday
    {
```

```
    updateTileAttr("precip_0_day", "Monday")      //today
    updateTileAttr("precip_1_day", "Sunday")    //yesteray
    updateTileAttr("precip_2_day", "Saturday")
    updateTileAttr("precip_3_day", "Friday")
    updateTileAttr("precip_4_day", "Thursday")
    updateTileAttr("precip_5_day", "Wednesday")
    updateTileAttr("precip_6_day", "Tuesday")
}
if (todayIndex == 3) //Tuesday
{
    updateTileAttr("precip_0_day", "Tuesday")      //today
    updateTileAttr("precip_1_day", "Monday")    //yesteray
    updateTileAttr("precip_2_day", "Sunday")
    updateTileAttr("precip_3_day", "Saturday")
    updateTileAttr("precip_4_day", "Friday")
    updateTileAttr("precip_5_day", "Thursday")
    updateTileAttr("precip_6_day", "Wednesday")
}
if (todayIndex == 4) //Wednesday
{
    updateTileAttr("precip_0_day", "Wednesday")      //today
    updateTileAttr("precip_1_day", "Tuesday")    //yesteray
    updateTileAttr("precip_2_day", "Monday")
    updateTileAttr("precip_3_day", "Sunday")
    updateTileAttr("precip_4_day", "Saturday")
    updateTileAttr("precip_5_day", "Friday")
    updateTileAttr("precip_6_day", "Thursday")
}
if (todayIndex == 5) //Thursday
{
    updateTileAttr("precip_0_day", "Thursday")      //today
    updateTileAttr("precip_1_day", "Wednesday")    //yesteray
    updateTileAttr("precip_2_day", "Tuesday")
    updateTileAttr("precip_3_day", "Monday")
    updateTileAttr("precip_4_day", "Sunday")
    updateTileAttr("precip_5_day", "Saturday")
    updateTileAttr("precip_6_day", "Friday")
}
if (todayIndex == 6) //Friday
{
    updateTileAttr("precip_0_day", "Friday")      //today
    updateTileAttr("precip_1_day", "Thursday")    //yesteray
    updateTileAttr("precip_2_day", "Wednesday")
    updateTileAttr("precip_3_day", "Tuesday")
```

```
        updateTileAttr("precip_4_day", "Monday")
        updateTileAttr("precip_5_day", "Sunday")
        updateTileAttr("precip_6_day", "Saturday")
    }
    if (todayIndex == 7) //Saturday
    {
        updateTileAttr("precip_0_day", "Saturday")      //today
        updateTileAttr("precip_1_day", "Friday")    //yesteray
        updateTileAttr("precip_2_day", "Thursday")
        updateTileAttr("precip_3_day", "Wednesday")
        updateTileAttr("precip_4_day", "Tuesday")
        updateTileAttr("precip_5_day", "Monday")
        updateTileAttr("precip_6_day", "Sunday")
    }
}

// HTML Weather Tiles Logic
def TodayWeatherTile() {
        if(txtEnable == true){log.debug "updateTile1 called"}           // log the data returned by
WU//
        htmlToday ="<div style='line-height:1.0; font-size:1em;'><br>Weather for
${device.currentValue('station_location')}<br></div>"
        htmlToday +="<div style='line-height:50%;'><br></div>"
        htmlToday +="<div style='line-height:1.0; font-size:0.75em; text-align: left;'><br>Forecast
for ${device.currentValue('today')}<br></div>"
        htmlToday +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>${device.currentValue('forecastToday')}<br></div>"
        updateTileAttr("htmlToday", "$htmlToday")
        if(txtEnable == true){log.debug "htmlToday contains ${htmlToday}"}           // log the data
returned by WU//
        if(txtEnable == true){log.debug "${htmlToday.length()}"}                // log the data
returned by WU//
        }

def TomorrowWeatherTile() {
        if(txtEnable == true){log.debug "updateTile2 called"}           // log the data returned by
WU//
        htmlTomorrow ="<div style='line-height:1.0; font-size:1em;'><br>Weather for
${device.currentValue('station_location')}<br></div>"
        htmlTomorrow +="<div style='line-height:50%;'><br></div>"
        htmlTomorrow +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>Forecast for ${device.currentValue('tomorrow')}<br></div>"
        htmlTomorrow +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>${device.currentValue('forecastTomorrow')}<br></div>"
```

```
        updateTileAttr("htmlTomorrow", "$htmlTomorrow")
        if(txtEnable == true){log.debug "htmlTomorrow contains ${htmlTomorrow}"}
// log the data returned by WU//
        if(txtEnable == true){log.debug "${htmlTomorrow.length()}"}            // log the data
returned by WU//
        }

def DayAfterTomorrowWeatherTile() {
        if(txtEnable == true){log.debug "updateTile3 called"}           // log the data returned by
WU//
        htmlDayAfterTomorrow ="<div style='line-height:1.0; font-size:1em;'><br>Weather for
${device.currentValue('station_location')}<br></div>"
        htmlDayAfterTomorrow +="<div style='line-height:50%;'><br></div>"
        htmlDayAfterTomorrow +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>Forecast for ${device.currentValue('dayAfterTomorrow')}<br></div>"
        htmlDayAfterTomorrow +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>${device.currentValue('forecastDayAfterTomorrow')}<br></div>"
        updateTileAttr("htmlDayAfterTomorrow", "$htmlDayAfterTomorrow")
        if(txtEnable == true){log.debug "htmlDayAfterTomorrow contains
${htmlDayAfterTomorrow}"}               // log the data returned by WU//
        if(txtEnable == true){log.debug "${htmlDayAfterTomorrow.length()}"}                    // log
the data returned by WU//
        }

def WeatherWarningTile() {
        if(txtEnable == true){log.debug "updateTile4 called"}           // log the data returned by
WU//
        htmlWarnings ="<div style='line-height:1.0; font-size:1em;'><br>Weather Warnings for
${device.currentValue('station_location')}<br></div>"
        htmlWarnings +="<div style='line-height:50%;'><br></div>"
        htmlWarnings +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>${device.currentValue('today')}: ${device.currentValue('weatherWarning')}<br></div>"
        htmlWarnings +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>${device.currentValue('tomorrow')}:
${device.currentValue('weatherWarningTomorrow')}<br></div>"
        htmlWarnings +="<div style='line-height:1.0; font-size:0.75em; text-align:
left;'><br>${device.currentValue('dayAfterTomorrow')}:
${device.currentValue('weatherWarningDATomorrow')}<br></div>"
        updateTileAttr("htmlWarnings", "$htmlWarnings")
        if(txtEnable == true){log.debug "htmlWarnings contains ${htmlWarnings}"}            // log
the data returned by WU//
        if(txtEnable == true){log.debug "${htmlWarnings.length()}"}            // log the data
returned by WU//
        }
```

```
// HTML 3 Day Forecast Tile Logic
def wu3dayfcst() {
    if(txtEnable == true){log.info "3-Day Forecast: $threedayforecast"}
    String sTD='<td>'
    String sTR='<tr><td>'
        String my3day

    String iconSunrise = '<img src=https://tinyurl.com/icnqz/wsr.png>'
    String iconSunset = '<img src=https://tinyurl.com/icnqz/wss.png>'
    String degreeSign = "°" + device.currentValue('TempUnit')
    String MeasureSign = device.currentValue('MeasureUnit')
    String sunriseLocal
    String strSunrise
    String sunsetLocal
    String strSunset
    String strRainToday = ''
    String lastPoll
    String lastObsDate
    String lastObsTime
    String s1stHeader
    String s2ndHeader
    String s3rdHeader
    String sStationName

    java.lang.Integer Tstart = "${device.currentValue('sunriseTimeLocal')}".indexOf('T')
    java.lang.Integer Tstop1 = "${device.currentValue('sunriseTimeLocal')}".indexOf(':', Tstart)
    java.lang.Integer Tstop2 = "${device.currentValue('sunriseTimeLocal')}".indexOf(':', Tstop1+1)
    sunriseLocal = "${device.currentValue('sunriseTimeLocal')}".substring(Tstart+1, Tstop2)
    strSunrise = "${convert24to12(sunriseLocal)}"
    if(txtEnable == true){log.info "Sunrise = $strSunrise"}

    Tstart = "${device.currentValue('sunsetTimeLocal')}".indexOf('T')
    Tstop1 = "${device.currentValue('sunriseTimeLocal')}".indexOf(':', Tstart)
    Tstop2 = "${device.currentValue('sunriseTimeLocal')}".indexOf(':', Tstop1+1)
    sunsetLocal = "${device.currentValue('sunsetTimeLocal')}".substring(Tstart+1, Tstop2)
    strSunset = "${convert24to12(sunsetLocal)}"
    if(txtEnable == true){log.info "Sunset = $strSunset"}

    String todayAlert = ''
    if ("${device.currentValue('weatherWarningCode')}" != "None")
    {
       todayAlert = " @"
    }
```

```
    String tomorrowAlert = ''
    if ("${device.currentValue('weatherWarningCodeTomorrow')}" != "None")
    {
        tomorrowAlert = " @"
    }
    String DAtomorrowAlert = ''
    if ("${device.currentValue('weatherWarningCodeDATomorrow')}" != "None")
    {
        DAtomorrowAlert = " @"
    }

    strRainNow = 'Rain '
    BigDecimal rainToday

    if ("${device.currentValue('precip_today')}" ? true : false)
    {
        rainToday = "${device.currentValue('precip_today')}".toBigDecimal()
        if(rainToday > 0.00)
        {
            strRainNow += rainToday.toString()
        }
        else
        {
            strRainNow += '0.0'
        }
    }
    if(txtEnable == true){log.info "rainToday = $rainToday"}

    def BigDecimal fctRainToday
    fctRainToday = GetAmountRain("${device.currentValue('fCstRainToday')}")
    if(fctRainToday > 0.00)
    {
        strRainToday = fctRainToday + " " + MeasureSign
    }
    else
    {
        strRainToday = 'None'
    }

    if(txtEnable == true){log.info "precipChanceToday =
${device.currentValue('precipChanceToday')}"}
    if(txtEnable == true){log.info "fCstRainToday = ${device.currentValue('fCstRainToday')}"}

    def BigDecimal fctRainTomorrow
```

```
    fctRainTomorrow = GetAmountRain("${device.currentValue('fCstRainTomorrow')}")
    if(fctRainTomorrow > 0.00)
    {
        strRainTomorrow = fctRainTomorrow + " " + MeasureSign
    }
    else
    {
        strRainTomorrow = 'None'
    }

    if(txtEnable == true){log.info "precipChanceTomorrow =
${device.currentValue('precipChanceTomorrow')}"}
    if(txtEnable == true){log.info "fCstRainTomorrow =
${device.currentValue('fCstRainTomorrow')}"}

    def BigDecimal fctRainDayAfter
    fctRainDayAfter = GetAmountRain("${device.currentValue('fCstRainDayAfterTomorrow')}")
    if(fctRainTomorrow > 0.00)
    {
        strRainDayAfter = fctRainDayAfter + " " + MeasureSign
    }
    else
    {
        strRainDayAfter = 'None'
    }

    if(txtEnable == true){log.info "precipChanceDayAfterTomorrow  =
${device.currentValue('precipChanceDayAfterTomorrow')}"}
    if(txtEnable == true){log.info "fCstRainDayAfterTomorrow =
${device.currentValue('fCstRainDayAfterTomorrow')}"}

    lastPoll = convert24to12("${device.currentValue('lastPollTime')}")

    lastObsDate = "${device.currentValue('observation_time')}"
    lastObsTime = lastObsDate.substring(10,16)
    lastObsTime = convert24to12(lastObsTime)

    sl = device.currentValue('station_location')
    // sn = device.currentValue('stationID')

    sLeftMarker = ''
    sRightMarker = ''
    sforecastLocation = ''
    if (gpsCoords ? true : false) // true if not null or zero
```

```
    {
        sforecastLocation = 'Custom GPS'
        sLeftMarker = "<"
        sRightMarker = ">"
        if(txtEnable == true){log.info "forecast for: Custom GPS: ${latitudeCust} **
${longitudeCust}"}
    }
    else
    {
        if (pollICAO ? true : false) // true if not null or zero
        {
            sforecastLocation = pollICAO
            sLeftMarker = "<"
            sRightMarker = ">"
            if(txtEnable == true){log.info "forecast for: ICAO: ${pollICAO}"}
        }
    }
    my3day = '<table>'  // style="font-size:90%"
    my3day += '<tr>'
    my3day += '<td>' + "$sl" //+ '</td>'  // style="border: 1px solid;"
    my3day += '<td style="min-width:5%">'//</td>'
    //my3day += '<td>' + sLeftMarker + '<br>' + "${device.currentValue("forecastTimeName")}" +
todayAlert + '</td>'
    my3day += '<td>' + sLeftMarker + "${device.currentValue("forecastTimeName")}" + todayAlert
//+ '</td>'
    my3day += '<td style="min-width:5%">'//</td>'
    my3day += '<td>' + sforecastLocation + "${device.currentValue('tomorrow')}" + tomorrowAlert
//+ '</td>'
    my3day += '<td style="min-width:5%">'//</td>'
    my3day += '<td>' + sRightMarker + "${device.currentValue('dayAfterTomorrow')}" +
DAtomorrowAlert //+ '</td>'
    my3day += sTR
    my3day += "Now " + "${device.currentValue('temperature')} " + degreeSign + "<br>Feels
${device.currentValue('feelsLike')} " + degreeSign + "<br>Humidity
${device.currentValue('humidity')}" + '%<br>' + strRainNow + '<p>'
    my3day += sTD
    my3day += sTD + "${device.currentValue('forecastTodayIcon')}"
    my3day += sTD
    my3day += sTD + "${device.currentValue('forecastTomorrowIcon')}"
        my3day += sTD
    my3day += sTD + "${device.currentValue('forecastDayAfterTomorrowIcon')}"
    my3day += sTR
        my3day += sTD
    my3day += sTD + "${device.currentValue('forecastPhraseToday')}"
```

```
    my3day += sTD
    my3day += sTD + "${device.currentValue('forecastPhraseTomorrow')}"
    my3day += sTD
    my3day += sTD + "${device.currentValue('forecastPhraseDayAfterTomorrow')}"
    my3day += sTR
    my3day += 'High/Low'
    my3day += sTD
    my3day += sTD + "${device.currentValue('temperatureMaxToday')}" + degreeSign + ' ' +
"${device.currentValue('temperatureMinToday')}" + degreeSign
    my3day += sTD
    my3day += sTD + "${device.currentValue('temperatureMaxTomorrow')}" + degreeSign + ' ' +
"${device.currentValue('temperatureMinTomorrow')}" + degreeSign
        my3day += sTD
    my3day += sTD + "${device.currentValue('temperatureMaxDayAfterTomorrow')}" +
degreeSign + ' ' + "${device.currentValue('temperatureMinDayAfterTomorrow')}" + degreeSign
    my3day += sTR
    my3day += 'Chance Precip'
    my3day += sTD
    my3day += sTD + "${device.currentValue('precipChanceToday')}" + "% ${strRainToday}"
    my3day += sTD
    my3day += sTD + "${device.currentValue('precipChanceTomorrow')}" + "%
${strRainTomorrow}"
    my3day += sTD
    my3day += sTD + "${device.currentValue('precipChanceDayAfterTomorrow')}" + "%
${strRainDayAfter}"
    my3day += '<tr><td colspan="7">'  //blank line

    if (rainhistory)
    {
        java.lang.Integer totalRainDays = device.currentValue('rainHistoryDays')
        if(totalRainDays == 7)
        {
            s1stHeader = "Last 3 Days Rain:"
            s2ndHeader = "Last 5 Days Rain:"
            s3rdHeader = "Last 7 Days Rain:"
        }
        else
        {
            s1stHeader = "Last 2 Days Rain:"
            s2ndHeader = "Last 4 Days Rain:"
            s3rdHeader = "Last 6 Days Rain:"
        }

        if(txtEnable == true){log.info "Rain History Days: $raindaysdisplay"}
```

```
        switch(raindaysdisplay) {
            case "No selection":
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + iconSunrise + strSunrise +
' ' + iconSunset + strSunset + ' @ ' + lastObsTime
                break;
            case "None":
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + iconSunrise + strSunrise +
' ' + iconSunset + strSunset + ' @ ' + lastObsTime
                break;
             case "Yesterday":
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + 'Yesterdays Rain:' + "
${device.currentValue('precip_Yesterday')} " + iconSunrise + strSunrise + ' ' + iconSunset +
strSunset + ' @ ' + lastObsTime
                break;
             case "Last 2/3-Days":
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + s1stHeader + "
${device.currentValue('precip_Last3Days')} " + iconSunrise + strSunrise + ' ' + iconSunset +
strSunset + ' @ ' + lastObsTime
                break;
             case "Last 4/5-Days":
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + s2ndHeader + "
${device.currentValue('precip_Last5Days')} " + iconSunrise + strSunrise + ' ' + iconSunset +
strSunset + ' @ ' + lastObsTime
                break;
             case "Last 6/7-Days":
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + s3rdHeader + "
${device.currentValue('precip_Last7Days')} " + iconSunrise + strSunrise + ' ' + iconSunset +
strSunset + ' @ ' + lastObsTime
                break;
             default:
                my3day += '<tr style="font-size:75%"> <td colspan="7">' + iconSunrise + strSunrise +
' ' + iconSunset + strSunset + ' @ ' + lastObsTime
                break;
        }
    }
    else
    {
        my3day += '<tr style="font-size:75%"> <td colspan="7">' + iconSunrise + strSunrise + ' ' +
iconSunset + strSunset + ' @ ' + lastObsTime
    }
    //display HTTP Warnngs
    if (state.HTTPErrorFlag == "Yes")
    {
        my3day += "(" + state.HTTPErrorTypes + ")"
```

```
    }
    //log.info 'before html3dayfcst length: (' + my3day.length() + ')'
    //only show my3day length if close to maximum of 1024
    lenmy3day = my3day.length() + 16
    if (lenmy3day > 1000) {my3day += ' - {'+ lenmy3day + '}'}

    my3day += '</table>'
    // show html length in device status
    state.HTTP3DayLength = lenmy3day

    if(txtEnable == true){log.info 'html3dayfcst length: (' + my3day.length() + ')'}
    //log.info ' final html3dayfcst length: (' + my3day.length() + ')'

    if(my3day.length() > 1024) {
            log.error('Too much data to display.</br></br>Current threedayfcstTile length (' +
my3day.length() + ') exceeds maximum tile length by ' + (my3day.length() - 1024).toString()  + '
characters.')
        my3day = '<table>' + sTR + 'Error! Tile greater than 1024 characters. ' + sTR +
my3day.length() + ' exceeds maximum tile length by ' + (my3day.length() - 1024).toString() + '
characters.' + sTR + 'Try reducing some 3 day forecast options.<br>' +
"${device.currentValue('htmlToday')}" + '</table>'
          }
    updateTileAttr('html3dayfcst', my3day.take(1024))
}

// HTML Rain Tiles Logic
def rainTile() {

          String htmlRainTile
    String s1stHeader
    String s2ndHeader
    String s3rdHeader

    java.lang.Integer totalRainDays = device.currentValue('rainHistoryDays')
    if(totalRainDays == 7)
    {
       s1stHeader = "Last 3 Days:"
       s2ndHeader = "Last 5 Days:"
       s3rdHeader = "Last 7 Days:"
    }
    else
    {
       s1stHeader = "Last 2 Days:"
       s2ndHeader = "Last 4 Days:"
```

```
        s3rdHeader = "Last 6 Days:"
    }

    htmlRainTile ="<table>"
    htmlRainTile +='<tr style="font-size:80%"><td>' +
"${device.currentValue('station_location')}<br>$totalRainDays Day Rain History"
    htmlRainTile +='<tr style="font-size:85%"><td>Yesterday:' + "
${device.currentValue('precip_Yesterday')}"
    htmlRainTile +='<tr style="font-size:85%"><td>' + s1stHeader + "
${device.currentValue('precip_Last3Days')}"
    htmlRainTile +='<tr style="font-size:85%"><td>' + s2ndHeader + "
${device.currentValue('precip_Last5Days')}"
    htmlRainTile +='<tr style="font-size:85%"><td>' + s3rdHeader + "
${device.currentValue('precip_Last7Days')}"
    htmlRainTile +='<tr style="font-size:85%"><td>    '  //blank line
          htmlRainTile +='</table>'

    updateTileAttr("htmlRainTile", "$htmlRainTile")
          if(txtEnable == true){log.debug "htmlRainTile contains ${htmlRainTile}"}                    //
log the data returned by WU//
    if(txtEnable == true){log.debug "htmlRainTile length: ${htmlRainTile.length()}"}            // log
the data returned by WU//

}

String convert24to12(String input )
{
    if ( input.indexOf(":") == -1 )
        throw ("")

    final String []temp = input.split(":")

    if ( temp.size() != 2 )
        throw ("")  // Add your throw code
                // This does not support time string with seconds

    java.lang.Integer h = temp[0] as int  // if h or m is not a number then exception
    java.lang.Integer m = temp[1] as int  // java.lang.NumberFormatException will be raised
                        // that can be cached or just terminate the program
    String dn

    if ( h < 0 || h > 23 )
        throw("")  // add your own throw code
                // hour can't be less than 0 or larger than 24
```

```
    if ( m < 0 || m > 59 )
        throw("")  // add your own throw code
                // minutes can't be less than 0 or larger than 60

    String mPad = ""
    String strM = m.toString()
    if ( strM.length() == 1 )
        mPad = "0" // minutes less the 1 char append a zero

    if ( h == 0 ){
        h = 12
        dn = "AM"
    } else if ( h == 12 ) {
        dn = "PM"
    } else if ( h > 12 ) {
        h = h - 12
        dn = "PM"
    } else {
        dn = "AM"
    }

    return h.toString() + ":" + mPad + m.toString() + " " + dn.toString()
}

def logsOff() {
            log.warn "Debug logging disabled..."
            device.updateSetting("txtEnable", [value: "false", type: "bool"])}
```