Console Log Outputs inserted for Weather Service Analysis

login as: pi

pi@192.168.178.38's password:

Linux raspberrypi 6.12.47+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.47-1+rpt1~bookworm (2025-09-16) aarch64

The programs included with the Debian GNU/Linux system are free software; the exact distribution terms for each program are described in the individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent permitted by applicable law.

Last login: Fri Jan 30 08:23:47 2026 from 192.168.178.56

pi@raspberrypi:~ $ cd ~/weather

pi@raspberrypi:~/weather $ cd src/routes/weatherProviders

pi@raspberrypi:~/weather/src/routes/weatherProviders $ nano local.ts

.....

```
    lastRainCount = isNaN(rainCount) ? lastRainCount : rainCount;

    // log Weather Observation captured from local datastream
    console.log( 'Observation captured from local datastream: ', obs );

    queue.unshift(obs);
```

.....

```
    protected async getWeatherDataInternal( coordinates: GeoCoordinates, pws: PWS | undefined ): Promise< WeatherData > {
        queue = queue.filter( obs => Math.floor(Date.now()/1000) - obs.timestamp < 24*60*60 );
```

```
        // log last 24 hours amount of Weather Data measurements
        console.log( 'Amount of Weather Data measurements (queue.length ): ', queue.length);
.....

        if (weather.precip > 0){
            weather.raining = true;
        }

        // log Weather Data derived from local datastream
        console.log( 'Weather Data derived from local datastream: ', weather );

        return weather;
.....

    protected async getWateringDataInternal( coordinates: GeoCoordinates, pws: PWS | undefined ): Promise< WateringData[] > {

        queue = queue.filter( obs => Math.floor(Date.now()/1000) - obs.timestamp < 24*60*60 );

        // log last 24 hours amount of Watering Data measurements
        console.log( 'Amount of Watering Data measurements (queue.length): ', queue.length );


.....

    const result: WateringData = {
            weatherProvider: "local",
            humidity: queue.reduce( ( sum, obs ) => !isNaN( obs.humidity ) && ++cHumidity ? sum + obs.humidity : sum, 0) / cHumidity,
            precip: queue.reduce( ( sum, obs ) => !isNaN( obs.precip ) && ++cPrecip ? sum + obs.precip : sum, 0),
            periodStartTime: Math.floor( queue[ queue.length - 1 ].timestamp ),
            minTemp: queue.reduce( (min, obs) => ( min > obs.temp ) ? obs.temp : min, Infinity ),
            maxTemp: queue.reduce( (max, obs) => ( max < obs.temp ) ? obs.temp : max, -Infinity ),
            minHumidity: queue.reduce( (min, obs) => ( min > obs.humidity ) ? obs.humidity : min, Infinity ),
            maxHumidity: queue.reduce( (max, obs) => ( max < obs.humidity ) ? obs.humidity : max, -Infinity ),
```

```
            solarRadiation: queue.reduce( (sum, obs) => !isNaN( obs.solarRadiation ) && ++cSolar ? sum + obs.solarRadiation : sum, 0) / cSolar,
            windSpeed: queue.reduce( (sum, obs) => !isNaN( obs.windSpeed ) && ++cWind ? sum + obs.windSpeed : sum, 0) / cWind
        };

            // log Watering Data derived from local datastream
            console.log( 'Watering Data derived from local datastream: ', result );

        if ( !( cTemp && cHumidity && cPrecip ) ||
            [ result.minTemp, result.minHumidity, -result.maxTemp, -result.maxHumidity ].includes( Infinity ) ||
            !( cSolar && cWind && cPrecip )) {


            // log weather data to show what kind of weather data is invalid or missing?
            console.log( "log: Invalid or missing weather data to support watering calculation from local PWS." );
            console.log( 'cTemp: ', cTemp );
            console.log( 'cHumidity: ', cHumidity );
            console.log( 'cPrecip: ', cPrecip );
            console.log( 'cSolar: ', cSolar );
            console.log( 'cWind: ', cWind );
            console.log( 'minTemp: ', result.minTemp );
            console.log( 'maxTemp: ', result.maxTemp );
            console.log( 'minHumidity: ', result.minHumidity );
            console.log( 'maxHumidity: ', result.maxHumidity );


            console.error( "There is insufficient data to support watering calculation from local PWS." );
            throw new CodedError( ErrorCode.InsufficientWeatherData );
        }
```

.....

pi@raspberrypi:~/weather/src/routes/weatherProviders $ cd ~/weather/src/routes/adjustmentMethods

```
pi@raspberrypi:~/weather/src/routes/adjustmentMethods $ ls

AdjustmentMethod.ts        EToAdjustmentMethod.ts     RainDelayAdjustmentMethod.ts
EToAdjustmentMethod.spec.ts  ManualAdjustmentMethod.ts  ZimmermanAdjustmentMethod.ts

pi@raspberrypi:~/weather/src/routes/adjustmentMethods $
```

```
......

    // Map data into proper format
    const rawData = wateringData.map(data => {
        return {
            wp: data.weatherProvider,
            h: data ? Math.round( data.humidity * 100) / 100 : null,
            p: data ? Math.round( data.precip * 100 ) / 100 : null,
            t: data ? Math.round( data.temp * 10 ) / 10 : null,
        };
    });

    // log Zimmerman Watering Data Adjustment Method Response
    console.log( 'Zimmerman Adjustment Method Watering Data: ', wateringData );

    for ( let i = 0; i < wateringData.length; i++ ) {
        // Check to make sure valid data exists for all factors
        if ( !validateValues( [ "temp", "humidity", "precip" ], wateringData[i] ) ) {
            // Default to a scale of 100% if fields are missing.
            throw new CodedError( ErrorCode.MissingWeatherField );
        }
    }
.......
```

```
pi@raspberrypi:~ $ cd ~/weather

pi@raspberrypi:~/weather $ cd src/routes/weatherProviders/
```

```
pi@raspberrypi:~/weather/src/routes/weatherProviders $ ls -l

total 88

-rw-r--r-- 1 pi pi  7977 Jan 25 22:10 AccuWeather.ts
-rw-r--r-- 1 pi pi 16986 Jan 25 22:10 Apple.ts
-rw-r--r-- 1 pi pi  8306 Jan 25 22:10 DWD.ts
-rw-r--r-- 1 pi pi  6300 Feb  7 07:03 local.ts
-rwxr-xr-x 1 pi pi  8988 Jan 25 22:10 OpenMeteo.ts
-rw-r--r-- 1 pi pi  5022 Jan 25 22:10 OWM.ts
-rw-r--r-- 1 pi pi  7829 Jan 25 22:10 PirateWeather.ts
-rw-r--r-- 1 pi pi  4096 Jan 25 22:10 WeatherProvider.ts
-rw-r--r-- 1 pi pi  7260 Jan 25 22:10 WUnderground.ts

pi@raspberrypi:~/weather/src/routes/weatherProviders $ nano WeatherProvider.ts

......

        const expiresAt = addDays(startOfDay(TZDate.tz(tz)), 1);

        // log WateringData expiry time
        console.log( ' WateringData expiry time: ', expiresAt );


        return this.wateringDataCache[key].get(() => this.getWateringDataInternal(coordinates, pws), expiresAt);

......

        const expiresAt = addHours(startOfDay(date), (Math.floor(date.getHours() / 6) + 1) * 6);

        // log WeatherData expiry time
        console.log( ' WeatherData expiry time: ', expiresAt );

        return this.weatherDataCache[key].get(() => this.getWeatherDataInternal(coordinates, pws), expiresAt);
```

.....

pi@raspberrypi:~/weather/src/routes/weatherProviders $ cd ~/weather

pi@raspberrypi:~/weather $ npm run build

> os-weather-service@3.0.2 build
> node build.mjs

pi@raspberrypi:~/weather $ npm run start

> os-weather-service@3.0.2 start
> node dist/index.cjs

OpenSprinkler Weather Service now listening on 0.0.0.0:3000
OpenSprinkler Weather Service now listening for local weather stream
Loaded baseline ETo data.

0|npm | Observation captured from local datastream: {
0|npm | timestamp: 1770703200,
0|npm | temp: 31.1,
0|npm | humidity: 95,
0|npm | windSpeed: 10.6,
0|npm | solarRadiation: NaN,
0|npm | precip: 0
0|npm | }
0|npm | Observation captured from local datastream: {
0|npm | timestamp: 1770703500,
0|npm | temp: 31.1,
0|npm | humidity: 95,
0|npm | windSpeed: 10.6,
0|npm | solarRadiation: NaN,

```
0|npm |   precip: 0
0|npm | }
 ^C
pi@raspberrypi:~/weather $
```