KerbalWeatherProject

Release 1.0.2

cmac994

Jan 31, 2022

CONTENTS

1 Guide **3**
1.1 Tutorial . . . 3
1.2 API Documentation . . . 5
1.3 License . . . 13
1.4 Need Help . . . 14

2 Indices and tables **15**

Index **17**

Documentation for KerbalWeatherProject's (KWP) climate and weather API.

See the tutorial for a guide on how to access and use the KWP API.

Click API documentation to view a list of api calls

CHAPTER

ONE

GUIDE

1.1 Tutorial

This tutorial demonstrates how to use the KerbalWeatherProject (KWP) API in a C# plugin for KSP.

Copy *KerbalWeatherProject.dll* to your KSP_x64_Data/Managed Folder in the KSP Game Directory.

Add *KerbalWeatherProject.dll* as a project reference.

- In Visual Studio this can be accomplished by clicking *Project* then *add Reference*. Browse and select *KerbalWeatherProject.dll*.

Include KWP as an assembly dependency in your project

```
>>> [assembly: KSPAssemblyDependency("KerbalWeatherProject", 1, 0)]
```

Open a class in which you'd like to reference the KWP API and add the following:

```
>>> using KerbalWeatherProject
```

Check to see if KWP is available

```
//Boolean to check for KWP in assembly
bool CheckKWP()
{
    try
    {
        //Define null type references
        Type weather = null;
        Type climate = null;
        //Sort through assemblies
        foreach (var assembly in AssemblyLoader.loadedAssemblies)
        {
            //Search for KWP
            if (assembly.name == "KerbalWeather_Project")
            {
                //Get assembly methods
                var types = assembly.assembly.GetExportedTypes();

                //Search for climate and weather api
                foreach (Type t in types)
                {
                    if (t.FullName.Equals("KerbalWeather_Project.climate_api"))
```

(continues on next page)

(continued from previous page)

```
                    {
                        climate = t;
                    }
                    if (t.FullName.Equals("KerbalWeather_Project.weather_api"))
                    {
                        weather = t;
                    }
                }
            }
        }

        //Ensure API exists
        if (weather == null || climate == null)
        {
            return false;
        }
        return true; // jump out
    }
    catch (Exception e)
    {
        Debug.LogError("[WxAPI]: unable to find KerbalWeather_Project. Exception thrown:
↪" + e.ToString());
    }
    return false;
}
```

Use the climate API to retrieve climatological data at a specific point in time and space.

```
//Set UT Time
epoch = 3600;

//Set position for climate API test
double mlat = 25.0; // 25 N
double mlng = -60.0; // 60 W
double malt = 5000; // 5-km ASL

double uwind_climo = climate_api.uwind(mlat, mlng, malt, epoch);
double vwind_climo = climate_api.vwind(mlat, mlng, malt, epoch);
double zwind_climo = climate_api.zwind(mlat, mlng, malt, epoch);

Debug.Log("Climatological U-Wind " + (malt / 1e3) + " km ASL at (" + mlat + "N, " + Math.
↪Abs(mlng) + "W) " + uwind_climo + " m/s");
Debug.Log("Climatological V-Wind " + (malt / 1e3) + " km ASL at (" + mlat + "N, " + Math.
↪Abs(mlng) + "W) " + vwind_climo + " m/s");
Debug.Log("Climatological Z-Wind " + (malt / 1e3) + " km ASL at (" + mlat + "N, " + Math.
↪Abs(mlng) + "W) " + zwind_climo + " m/s");
```

```
Climatological U-Wind 5 km ASL at (25N, 60W) 21.4549880545088 m/s
Climatological V-Wind 5 km ASL at (25N, 60W) -1.55983404053068 m/s
Climatological Z-Wind 5 km ASL at (25N, 60W) -0.0169466099952593 m/s
```

Use the weather API to retrieve point weather data at a given time and height (above each launch site).

```
//Altitude above sea level
double altitude = 0.0;

//Get list of launch sites with weather data
List<string> lsites = weather_api.lsites;

//Loop through launch sites
for (int l = 0; l < 3; l++)
{

        //Set launch site
        lsite = lsites[l];

        //Read weather data from launch site
        weather_api.set_datasource(lsite);

        //Get temperature data for launch site
        double tmp_ls = weather_api.temperature(altitude, epoch);
        Debug.Log("Temperature at " + lsite + " "+altitude+" m ASL: " + tmp_ls+" K");
}
```

```
Temperature at KSC: 300.649475097656 K
Temperature at DLS: 288.496887207031 K
Temperature at WLS: 243.553863525391 K
```

Note: If using the Lite version of KerbalWeatherProject replace *KerbalWeatherProject* with *KerbalWeatherProject_Lite* for all references above.

1.2 API Documentation

1.2.1 Climate API

Utility

Variable lists.

climate_api.**get_vars3D**()
    Returns (Dictionary): 3D atmospheric variables accessible with the KWP climate API.

    (Key = variable name, Value = variable index)

climate_api.**get_vars2D**()
    Returns (Dictionary) 2D atmospheric variables accessible with the KWP climate API.

    (Key = variable name, Value = variable index)

Wind

Retrieve atmospheric wind data

`climate_api.`**`uwind`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): zonal-wind component (m/s). Wind velocity in east-west direction.

`climate_api.`**`vwind`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): meridional wind component (m/s). Wind velocity in north-south direction.

`climate_api.`**`zwind`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): vertical wind component (m/s). wind velocity in up-down direction.

Ambient Conditions

Retrieve column (3D) atmospheric variables

`climate_api.`**`pressure`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): air pressure (Pa)

`climate_api.`**`temperature`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees

- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): air temperature (K)

`climate_api.`**`relative_humidity`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): relative_humidity (%)

`climate_api.`**`cloud_cover`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): cloud_cover (%) - above altitude. Percentage of sky above covered by clouds.

`climate_api.`**`visibility`**`(`*latitude*, *longitude*, *altitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): visibility (km). Estimate of visibility derived from humidity, cloud cover, and precipitation rate.

Surface Conditions

Retrieve surface (2D) atmospheric variables

`climate_api.`**`OLR`**`(`*latitude*, *longitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- ut (double) - universal time in seconds (time since game began)

Returns (double): outgoing longwave radiation (w/m^2). Returned from IR satellite imagery and used to view cloud cover in the absence of visible light.

`climate_api.`**`total_cloud_cover`**`(`*latitude*, *longitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- ut (double) - universal time in seconds (time since game began)

Returns (double): total cloud cover (%). Percentage of sky covered by clouds.

`climate_api.`**`precipitable_water`**`(`*latitude*, *longitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- ut (double) - universal time in seconds (time since game began)

Returns (double): precipitable water (mm). Amount of liquid water produced by the condensation of all available water vapor in the atmospheric column above a given point. Estimates the moisture content of the atmosphere.

`climate_api.`**`prate`**`(`*latitude*, *longitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- ut (double) - universal time in seconds (time since game began)

Returns (double): precipitation rate (mm/hr). Liquid water equivalent precipitation rate, derived from convective and stratiform precipitation totals.

`climate_api.`**`mslp`**`(`*latitude*, *longitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- ut (double) - universal time in seconds (time since game began)

Returns (double): mean sea level pressure (Pa). Pressure, reduced to sea level, by accounting for the elevation of terrain and diurnal variations in temperature.

`climate_api.`**`sst`**`(`*latitude*, *longitude*, *ut*`)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees
- ut (double) - universal time in seconds (time since game began)

Returns (double): skin surface temperature (K). On land = land surface temperature. On water = sea surface temperature (SST).

Derivatives

Derive variables from climate API calls above.

`climate_api.`**`density`**`(`*pressure*, *temperature*`)`

Parameters

- pressure (double) - air pressure (Pa)
- temperature (double) - air temperature (K)

Returns (double): air density (kg/m^3)

`climate_api.`**`wspd`**`(`*uwind*, *vwind*, *zwind*`)`

Parameters

- uwind (double) - zonal wind component (m/s)
- vwind (double) - meridional wind component (m/s)
- zwind (double) - vertical wind component (m/s)

Returns (double): wind speed (m/s)

`climate_api.`**`wdir_degrees`**`(`*uwind*, *vwind*`)`

Parameters

- uwind (double) - zonal wind component (m/s)
- vwind (double) - meridional wind component (m/s)

Returns (double): wind direction (degrees). Direction in which the wind is coming from (e.g. 45 or 225).

`climate_api.`**`wdir_cardinal`**`(`*wdir_degrees*`)`

Parameters

- wdir_degrees (double) - wind direction (degrees)

Returns (string): cardinal wind direction. Direction in which the wind is coming from (e.g. NE or SW)

`climate_api.`**`cloud_top_temps`**`(`*olr*`)`

Parameters

- olr (double) - outgoing longwave radiation (W/m^2)

Returns (double): cloud top temperatures (K). Cloud top temperature. If skies are clear this is an estimate of the land/sea surface temperature.

1.2.2 Weather API

Utility

List of available launch sites and atmospheric variables

lsites (List<string>)

- list of available launch sites (three letter abbreviations)

lsites_name (List<string>)

- list of available launch sites (full names)

lsites_lat (List<double>)

- list of launch site latitudes

lsites_lng (List<double>)

- list of launch site longitudes

`weather_api.set_datasource(launch_site)`

Parameters

- launch_site (string) - three letter launch site abbreviation (e.g. KSC)

Returns (void): Reads weather data, at the specified launch site, into memory.

`weather_api.get_nearest_lsite_idx(latitude, longitude)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees

Returns (int): Index of nearest launch site in list (int).

`weather_api.get_nearest_lsite(latitude, longitude)`

Parameters

- latitude (double) - decimal degrees
- longitude (double) - decimal degrees

Returns (string): Nearest launch site.

`weather_api.get_vars3D()`

Returns (Dictionary): 3D atmospheric variables accessible with the KWP weather API.

(Key = variable name, Value = variable index)

`weather_api.get_vars2D()`

Returns (Dictionary) 2D atmospheric variables accessible with the KWP weather API.

(Key = variable name, Value = variable index)

Wind

Retrieve atmospheric wind data

`weather_api.uwind(altitude, ut)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): zonal-wind component (m/s). Wind velocity in east-west direction.

`weather_api.vwind(altitude, ut)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): meridional wind component (m/s). Wind velocity in north-south direction.

`weather_api.zwind(altitude, ut)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): vertical wind component (m/s). wind velocity in up-down direction.

Ambient Conditions

Retrieve column (3D) atmospheric variables

`weather_api.`**`pressure`**`(`*altitude*, *ut*`)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): air pressure (Pa)

`weather_api.`**`temperature`**`(`*altitude*, *ut*`)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): air temperature (K)

`weather_api.`**`relative_humidity`**`(`*altitude*, *ut*`)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): relative_humidity (%)

`weather_api.`**`cloud_cover`**`(`*altitude*, *ut*`)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): cloud_cover (%) - above altitude. Percentage of sky above covered by clouds.

`weather_api.`**`visibility`**`(`*altitude*, *ut*`)`

Parameters

- altitude (double) - meters above sea level
- ut (double) - universal time in seconds (time since game began)

Returns (double): visibility (km). Estimate of visibility derived from humidity, cloud cover, and precipitation rate.

Surface Conditions

Retrieve surface (2D) atmospheric variables

`weather_api.`**`OLR`**`(`*ut*`)`

Parameters

- ut (double) - universal time in seconds (time since game began)

Returns (double): outgoing longwave radiation (w/m^2). Returned from IR satellite imagery and used to view cloud cover in the absence of visible light.

`weather_api.`**`total_cloud_cover`**`(`*ut*`)`

Parameters

- ut (double) - universal time in seconds (time since game began)

Returns (double): total cloud cover (%). Percentage of sky covered by clouds.

`weather_api.`**`precipitable_water`**`(`*ut*`)`

Parameters

- ut (double) - universal time in seconds (time since game began)

Returns (double): precipitable water (mm). Amount of liquid water produced by the condensation of all available water vapor in the atmospheric column above a given point. Estimates the moisture content of the atmosphere.

`weather_api.`**`prate`**`(`*ut*`)`

Parameters

- ut (double) - universal time in seconds (time since game began)

Returns (double): precipitation rate (mm/hr). Liquid water equivalent precipitation rate, derived from convective and stratiform precipitation totals.

`weather_api.`**`mslp`**`(`*ut*`)`

Parameters

- ut (double) - universal time in seconds (time since game began)

Returns (double): mean sea level pressure (Pa). Pressure, reduced to sea level, by accounting for the elevation of terrain and diurnal variations in temperature.

`weather_api.`**`sst`**`(`*ut*`)`

Parameters

- ut(double) - universal time in seconds (time since game began)

Returns (double): skin surface temperature (K). On land = land surface temperature. On water = sea surface temperature (SST).

Derivatives

Derive variables from weather API calls above.

`weather_api.`**`density`**`(`*pressure*, *temperature*`)`

Parameters

- pressure (double) - air pressure (Pa)
- temperature (double) - air temperature (K)

Returns (double): air density (kg/m^3)

`weather_api.wspd(uwind, vwind, zwind)`

Parameters

- uwind (double) - zonal wind component (m/s)
- vwind (double) - meridional wind component (m/s)
- zwind (double) - vertical wind component (m/s)

Returns (double): wind speed (m/s)

`weather_api.wdir_degrees(uwind, vwind)`

Parameters

- uwind (double) - zonal wind component (m/s)
- vwind (double) - meridional wind component (m/s)

Returns (double): wind direction (degrees). Direction in which the wind is coming from (e.g. 45 or 225).

`weather_api.wdir_cardinal(wdir_degrees)`

Parameters

- wdir_degrees (double) - wind direction (degrees)

Returns (string): cardinal wind direction. Direction in which the wind is coming from (e.g. NE or SW)

`weather_api.cloud_top_temps(olr)`

Parameters

- olr (double) - outgoing longwave radiation (W/m^2)

Returns (double): cloud top temperatures (K). Cloud top temperature. If skies are clear this is an estimate of the land/sea surface temperature.

1.3 License

1.4 Need Help

If you’re having trouble please post to the KSP Forum Page or submit an issue on GitHub.

CHAPTER

TWO

INDICES AND TABLES

- genindex
- search

INDEX

B

built-in function
- climate_api.cloud_cover(), 7
- climate_api.cloud_top_temps(), 9
- climate_api.density(), 9
- climate_api.get_vars2D(), 5
- climate_api.get_vars3D(), 5
- climate_api.mslp(), 8
- climate_api.OLR(), 7
- climate_api.prate(), 8
- climate_api.precipitable_water(), 8
- climate_api.pressure(), 6
- climate_api.relative_humidity(), 7
- climate_api.sst(), 8
- climate_api.temperature(), 6
- climate_api.total_cloud_cover(), 7
- climate_api.uwind(), 6
- climate_api.visibility(), 7
- climate_api.vwind(), 6
- climate_api.wdir_cardinal(), 9
- climate_api.wdir_degrees(), 9
- climate_api.wspd(), 9
- climate_api.zwind(), 6
- weather_api.cloud_cover(), 11
- weather_api.cloud_top_temps(), 13
- weather_api.density(), 12
- weather_api.get_nearest_lsite(), 10
- weather_api.get_nearest_lsite_idx(), 10
- weather_api.get_vars2D(), 10
- weather_api.get_vars3D(), 10
- weather_api.mslp(), 12
- weather_api.OLR(), 12
- weather_api.prate(), 12
- weather_api.precipitable_water(), 12
- weather_api.pressure(), 11
- weather_api.relative_humidity(), 11
- weather_api.set_datasource(), 10
- weather_api.sst(), 12
- weather_api.temperature(), 11
- weather_api.total_cloud_cover(), 12
- weather_api.uwind(), 10
- weather_api.visibility(), 11
- weather_api.vwind(), 10
- weather_api.wdir_cardinal(), 13
- weather_api.wdir_degrees(), 13
- weather_api.wspd(), 13
- weather_api.zwind(), 10

C

climate_api.cloud_cover()
- built-in function, 7

climate_api.cloud_top_temps()
- built-in function, 9

climate_api.density()
- built-in function, 9

climate_api.get_vars2D()
- built-in function, 5

climate_api.get_vars3D()
- built-in function, 5

climate_api.mslp()
- built-in function, 8

climate_api.OLR()
- built-in function, 7

climate_api.prate()
- built-in function, 8

climate_api.precipitable_water()
- built-in function, 8

climate_api.pressure()
- built-in function, 6

climate_api.relative_humidity()
- built-in function, 7

climate_api.sst()
- built-in function, 8

climate_api.temperature()
- built-in function, 6

climate_api.total_cloud_cover()
- built-in function, 7

climate_api.uwind()
- built-in function, 6

climate_api.visibility()
- built-in function, 7

climate_api.vwind()
- built-in function, 6

climate_api.wdir_cardinal()

built-in function, 9
climate_api.wdir_degrees()
built-in function, 9
climate_api.wspd()
built-in function, 9
climate_api.zwind()
built-in function, 6

W

weather_api.cloud_cover()
built-in function, 11
weather_api.cloud_top_temps()
built-in function, 13
weather_api.density()
built-in function, 12
weather_api.get_nearest_lsite()
built-in function, 10
weather_api.get_nearest_lsite_idx()
built-in function, 10
weather_api.get_vars2D()
built-in function, 10
weather_api.get_vars3D()
built-in function, 10
weather_api.mslp()
built-in function, 12
weather_api.OLR()
built-in function, 12
weather_api.prate()
built-in function, 12
weather_api.precipitable_water()
built-in function, 12
weather_api.pressure()
built-in function, 11
weather_api.relative_humidity()
built-in function, 11
weather_api.set_datasource()
built-in function, 10
weather_api.sst()
built-in function, 12
weather_api.temperature()
built-in function, 11
weather_api.total_cloud_cover()
built-in function, 12
weather_api.uwind()
built-in function, 10
weather_api.visibility()
built-in function, 11
weather_api.vwind()
built-in function, 10
weather_api.wdir_cardinal()
built-in function, 13
weather_api.wdir_degrees()
built-in function, 13
weather_api.wspd()
built-in function, 13
weather_api.zwind()
built-in function, 10