

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using isci.clima.et;

namespace isci.clima.web.et
{
    /// <summary>
    /// The web service ET makes available some methods of the
    /// component ET
    /// </summary>
    [WebService (Namespace="http://www.sipeaa.it/asp/webservices/et/")]
    public class ISCI_CLIMA_ServiceET : System.Web.Services.WebService
    {
        public ISCI_CLIMA_ServiceET()
        {
            //CODEGEN: This call is required by the ASP.NET Web Services Designer
            InitializeComponent();
        }

        #region Component Designer generated code

        //Required by the Web Services Designer
        private IContainer components = null;

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {

```

```

        components.Dispose();
    }
    base.Dispose(disposing);
}

#endregion

[WebMethod]
public double VaporPressureDeficit(double maxAirTemperature, double minAirTemperature,
                                   double maxRelativeAirHumidity, double minRelativeAirHumidity)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ETIn.minAirTemperature = minAirTemperature;
    ETIn.maxRelativeAirHumidity = maxRelativeAirHumidity;
    ETIn.minRelativeAirHumidity = minRelativeAirHumidity;
    ET et = new ET();
    et.VaporPressureDeficit(ref ETIn, ref ETOut);
    return Math.Round(ETOut.vaporPressureDeficit, 2);
}

[WebMethod]
public double SaturatedVaporPressure(double maxAirTemperature)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ET et = new ET();
    et.SlopeSaturatedVaporPressure(ref ETIn, ref ETOut);
    return Math.Round(ETOut.saturatedVaporPressure, 2);
}

[WebMethod]
public double SlopeSaturatedVaporPressure(double maxAirTemperature, double minAirTemperature)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ETIn.minAirTemperature = minAirTemperature;
    ET et = new ET();
    et.SlopeSaturatedVaporPressure(ref ETIn, ref ETOut);
    return Math.Round(ETOut.slopeSaturatedVaporPressure, 4);
}

[WebMethod]

```

```

public double ActualVaporPressure(double maxAirTemperature, double minAirTemperature,
    double maxRelativeAirHumidity, double minRelativeAirHumidity)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ETIn.minAirTemperature = minAirTemperature;
    ETIn.maxRelativeAirHumidity = maxRelativeAirHumidity;
    ETIn.minRelativeAirHumidity = minRelativeAirHumidity;
    ET et = new ET();
    et.ActualVaporPressure(ref ETIn, ref ETOut);
    return Math.Round(ETOut.actualVaporPressure,2);
}

```

[WebMethod]

```

public double IsothermallongWaveNetRadiation(double maxAirTemperature,
    double minAirTemperature, double potentialSolarRadiation,
    double globalSolarRadiation, double clearSkyTransmissivity)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ETIn.minAirTemperature = minAirTemperature;
    ETIn.potentialSolarRadiation = potentialSolarRadiation;
    ETIn.globalSolarRadiation = globalSolarRadiation;
    ETIn.clearSkyTransmissivity = clearSkyTransmissivity;
    ET et = new ET();
    et.IsothermallongWaveNetRadiation(ref ETIn, ref ETOut);
    return Math.Round(ETOut.isothermallongWaveNetRadiation,2);
}

```

[WebMethod]

```

public double NetRadiation(double maxAirTemperature, double minAirTemperature,
    double potentialSolarRadiation, double globalSolarRadiation,
    double clearSkyTransmissivity)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ETIn.minAirTemperature = minAirTemperature;
    ETIn.potentialSolarRadiation = potentialSolarRadiation;
    ETIn.globalSolarRadiation = globalSolarRadiation;
    ETIn.clearSkyTransmissivity = clearSkyTransmissivity;
    ET et = new ET();
}

```

```

        et.NetRadiation(ref ETIn, ref ETOut);
        return Math.Round(ETOut.netRadiation,2);
    }

    [WebMethod]
    public double ReferenceETPenmanMonteith(double maxAirTemperature, double minAirTemperature,
        double potentialSolarRadiation, double globalSolarRadiation,
        double clearSkyTransmissivity, double windSpeed, double windMeasurementHeight,
        double maxRelativeAirHumidity, double minRelativeAirHumidity)
    {
        ETInput ETIn = new ETInput();
        ETOutput ETOut = new ETOutput();
        ETIn.maxAirTemperature = maxAirTemperature;
        ETIn.minAirTemperature = minAirTemperature;
        ETIn.potentialSolarRadiation = potentialSolarRadiation;
        ETIn.globalSolarRadiation = globalSolarRadiation;
        ETIn.clearSkyTransmissivity = clearSkyTransmissivity;
        ETIn.windMeasurementHeight = windMeasurementHeight;
        ETIn.windSpeed = windSpeed;
        ETIn.maxRelativeAirHumidity = maxRelativeAirHumidity;
        ETIn.minRelativeAirHumidity = minRelativeAirHumidity;
        PenmanMonteith m = new PenmanMonteith();
        ETIn.methodET = m;
        ET et = new ET();
        et.ReferenceET(ref ETIn, ref ETOut, false);
        return Math.Round(ETOut.referencePET,2);
    }

    [WebMethod]
    public double[] ReferenceETPenmanMonteithHourly(double[] hourlyAirTemperature,
        double[] hourlyPotentialSolarRadiation, double[] hourlyGlobalSolarRadiation,
        double clearSkyTransmissivity, double[] hourlyWindSpeed,
        double windMeasurementHeight, double[] hourlyRelativeAirHumidity,
        double potentialSolarRadiation)
    {
        ETInput ETIn = new ETInput();
        ETOutput ETOut = new ETOutput();
        int h = 0;
        for (h=0; h<24; h++)
        {
            ETIn.hourlyAirTemperature[h] = hourlyAirTemperature[h];
            ETIn.hourlyGlobalSolarRadiation[h] = hourlyGlobalSolarRadiation[h];
            ETIn.hourlyPotentialSolarRadiation[h] = hourlyPotentialSolarRadiation[h];
            ETIn.hourlyRelativeAirHumidity[h] = hourlyRelativeAirHumidity[h];
        }
    }

```

```

        ETIn.hourlyWindSpeed[h] = hourlyWindSpeed[h];
    }
    ETIn.clearSkyTransmissivity = clearSkyTransmissivity;
    ETIn.windMeasurementHeight = windMeasurementHeight;
    ETIn.potentialSolarRadiation = potentialSolarRadiation;
    PenmanMonteithHourly m = new PenmanMonteithHourly();
    ETIn.methodET = m;
    ET et = new ET();
    et.ReferenceET(ref ETIn, ref ETOut, false);
    double[] hET = new double[24];
    hET = ETOut.hourlyReferencePET;
    return hET;
}

[WebMethod]
public double[] VaporPressureDeficitHourly(double[] hourlyAirTemperature,
    double[] hourlyPotentialSolarRadiation, double[] hourlyGlobalSolarRadiation,
    double clearSkyTransmissivity, double[] hourlyWindSpeed,
    double windMeasurementHeight, double[] hourlyRelativeAirHumidity,
    double potentialSolarRadiation)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    double[] hET = new double[24];
    int h = 0;
    for (h=0; h<24; h++)
    {
        ETIn.hourlyAirTemperature[h] = hourlyAirTemperature[h];
        ETIn.hourlyGlobalSolarRadiation[h] = hourlyGlobalSolarRadiation[h];
        ETIn.hourlyPotentialSolarRadiation[h] = hourlyPotentialSolarRadiation[h];
        ETIn.hourlyRelativeAirHumidity[h] = hourlyRelativeAirHumidity[h];
        ETIn.hourlyWindSpeed[h] = hourlyWindSpeed[h];
    }
    ETIn.clearSkyTransmissivity = clearSkyTransmissivity;
    ETIn.windMeasurementHeight = windMeasurementHeight;
    ETIn.potentialSolarRadiation = potentialSolarRadiation;
    PenmanMonteithHourly m = new PenmanMonteithHourly();
    ETIn.methodET = m;
    ET et = new ET();
    et.ReferenceET(ref ETIn, ref ETOut, false);
    double[] hVPD = new double[24];
    hVPD = ETOut.hourlyVaporPressureDeficit;
    return hVPD;
}

```

```
[WebMethod]
public double ReferenceETHargreaves(double maxAirTemperature,
    double minAirTemperature, double potentialSolarRadiation,
    double slopeHargreaves, double interceptHargreaves)
{
    ETInput ETIn = new ETInput();
    ETOutput ETOut = new ETOutput();
    ETIn.maxAirTemperature = maxAirTemperature;
    ETIn.minAirTemperature = minAirTemperature;
    ETIn.potentialSolarRadiation = potentialSolarRadiation;
    ETIn.slopeHargreaves = slopeHargreaves;
    ETIn.interceptHargreaves = interceptHargreaves;
    Hargreaves m = new Hargreaves();
    ETIn.methodET = m;
    ET et = new ET();
    et.ReferenceET(ref ETIn, ref ETOut, false);
    return Math.Round(ETOut.referencePET,2);
}
}
}
```