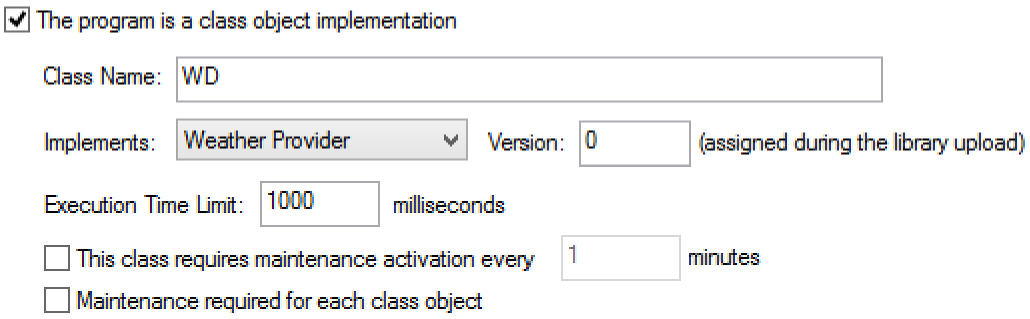
HCA Tech Note 606

Weather Classes

To facilitate users getting weather data from a variety of sources containing a variety of data in addition to the built-in weather support, a new weather provider can be supported by creating a weather class - available in HCA 15.1 and later.

When creating the class program, it is marked as implementing a “Weather Provider”.



☑ The program is a class object implementation

Class Name: WD

Implements: Weather Provider Version: 0 (assigned during the library upload)

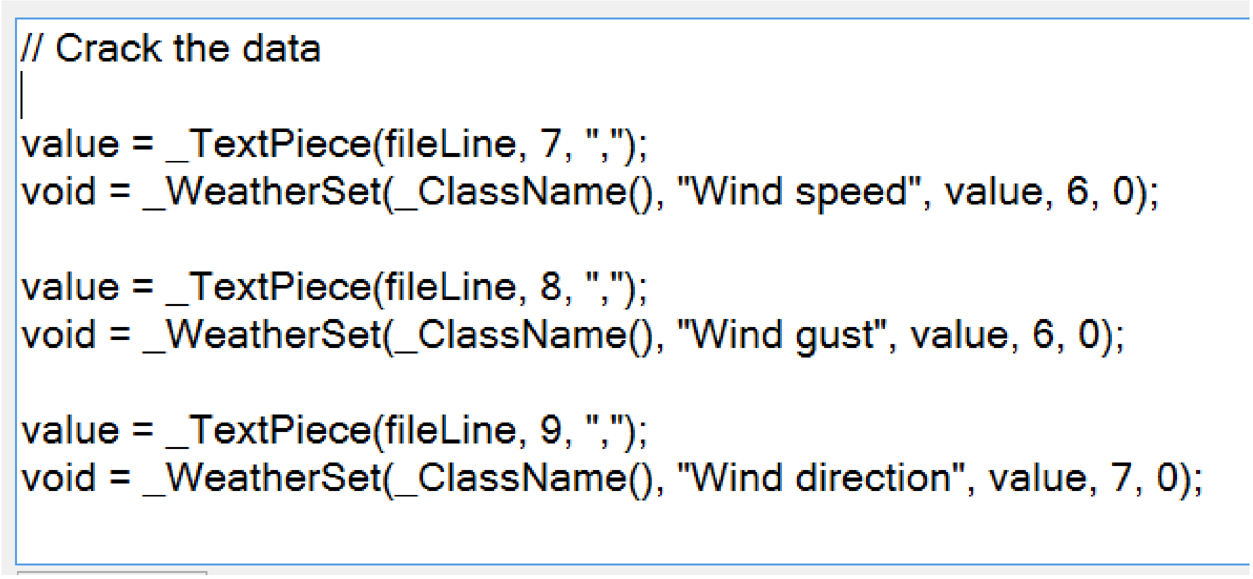
Execution Time Limit: 1000 milliseconds

☐ This class requires maintenance activation every 1 minutes

☐ Maintenance required for each class object

The class can be named whatever is appropriate and its implementation is determined by the class creator.

As an example, here are the contents of a Compute element used in the “Weather Display” class. Weather Display is a program that interfaces directly with a variety of weather stations and periodically exports data to a file called scriptlog.txt. After the class program opens the file (using the _FileOpen function) and reads the header and data lines (uses the _ReadString function), it extracts the data from the known file format.



```
// Crack the data

value = _TextPiece(fileLine, 7, ",");
void = _WeatherSet(_ClassName(), "Wind speed", value, 6, 0);

value = _TextPiece(fileLine, 8, ",");
void = _WeatherSet(_ClassName(), "Wind gust", value, 6, 0);

value = _TextPiece(fileLine, 9, ",");
void = _WeatherSet(_ClassName(), "Wind direction", value, 7, 0);
```

It stores the data into HCA using the _WeatherSet function. That takes 5 arguments:

1. The class name
2. The name of the data item
3. The value of the data item
4. The type of data item
5. The units the data value is in

The “type” is an encoding from this table:

Sensor Type	Code
0	Other
1	Temperature
2	Humidity (percent)
3	Pressure
4	Precipitation Intensity (amount / time period)
5	Precipitation Amount (amount)
6	Wind speed
7	Wind direction (0-360)
8	Cloud Cover (percent)
9	UV Index
10	Solar Radiation
11	Visibility (distance)

The “units” argument is an encoding from this table:

Sensor Type	Encoding	Units
Pressure Units	0	Inches
	1	Millimeters
	2	Millibars
	3	Hecto-pascals
Temperature Units	0	Fahrenheit
	1	Celsius
Wind speed units	0	MPH
	1	Knots
	2	KM / Hour
	3	Meters / second
Distance units	0	Miles
	1	Kilometers
Precipitation Intensity	0	In / hour
	1	Mm / hour
Precipitation	0	Inches
	1	Millimeters

The reason for having the data type and units arguments is for display and checking for reasonable data values used in the Weather VP element. If the data doesn’t fit any of the categories given above, then the “other” type (encoding = 0) is used.

The key points about Weather Classes:

1. HCA isn't defining the names of any of the weather items. The class writer is free to name the items whatever they want.

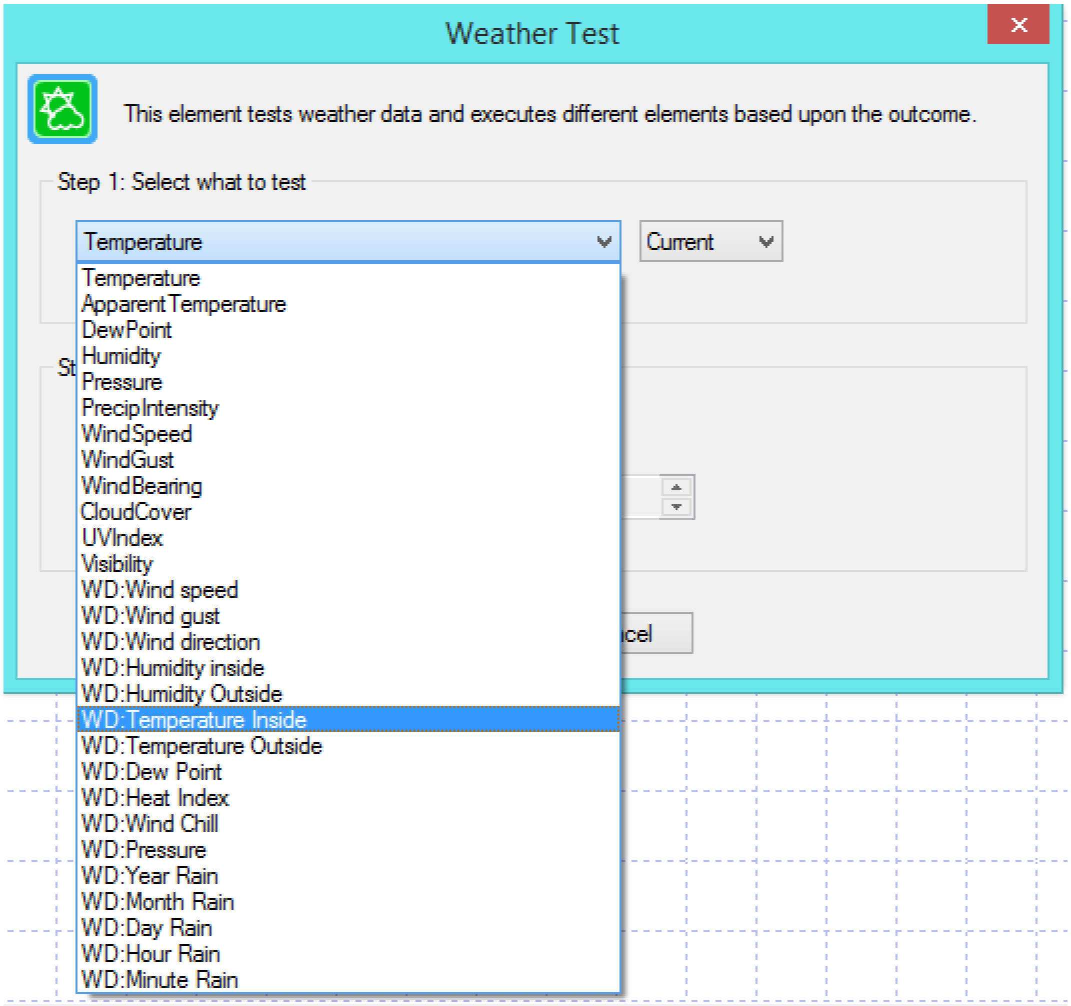
2. A class could offer as few or as many items as they want. There are no limits.

3. The class can access the provider in any way it wants. Read a file, do a network transaction, HTTP, whatever.

4. The class program is just a program and can be run as needed by whatever the user wants. If it is reading a file produced by another program, it could be run every few minutes. Or if it is using an online pay system maybe only once every few hours. That is up to the class creator. The suggestion is to use the "auto restart" facilities in the program's properties to cause the program to run every so many minutes.

The weather element shows both the standard Dark Sky items, from the built-in weather provider implementation, plus all from any of the weather classes that have been added to the design. For example, once the Weather Display ("WD") class was added to the design, the Weather element now allows access and testing of the new values.

Note that each weather provider class is shown with the items it provides as <class name>:<item name>. This means, that a design can utilize simultaneously as many weather providers as needed. This makes it possible to use the best features of each as needed.



Weather Test

This element tests weather data and executes different elements based upon the outcome.

Step 1: Select what to test

Temperature

Current

The built-in Dark Sky items are shown with their name with no prefix. The class items are shown as <Class name>:<Item name>

When using the _WeatherGet function, naming is done in the same manner:

X = _WeatherGet(“WD: Wind speed”);

All a user must to do is to import the class from the library and add it to their design. They may have to modify the class to change any settings specific to the installation. In the Weather Display class, the location of the “scriptlog.txt” file must be set as needed. In other cases, a global variable might need to be updated.

Why use Weather Classes?

The goal is to eliminate the need for each new weather provider to be built-in to HCA and to make a system that encompasses them all. For example, the old pre-V15 weather test element and weather trigger were difficult to maintain. Not all data was available from all the providers. It was not extensible.

While the built-in Weather Provider Dark Sky is good, there are other providers out there that may be better in some ways. The class model allows them to be added to HCA without needing to change HCA.

##end#