

Preliminary

Flash API Manual

Describes the Application Program Interface (API) for each Phidget device. The API is meant to be called from ActionScript, and the code examples are provided in ActionScript.

How to use Phidgets

Phidgets are an easy to use set of building blocks for low cost sensing and control from your PC. Using the Universal Serial Bus (USB) as the basis for all Phidgets, the complexity is managed behind this easy to use and robust Application Program Interface (API) library.

The library is a network library – the computer with the phidgets attached must be running the PhidgetWebService; even if the computer with the phidgets is also running the Shockwave application. The computer running the Shockwave application does not require a USB Host – just network access. This allows this library to be used on devices that can play Shockwave, but cannot necessarily have Phidgets plugged into them.

Installing the Library

To access the Phidget Actionsript software components, you must first install the library on your computer:

- To install the library go to www.Phidgets.com >> Downloads >> Release
- Select the Phidget Flash Library file.

A dialog box will appear, asking if you would like to open the file or save it to your computer. .

Now, to use Phidgets from the Macromedeia Flash, just copy the actionsript files you need into your project folder.

Running the PhidgetWebService

The PhidgetWebService must be running on the computer that has the Phidgets attached. PhidgetWebService.exe is installed into C:\Program Files\Phidgets by Phidget.MSI, so it should already on your computer. The PhidgetWebService can be run as a service and controlled by the PhidgetWebService Manager found in the Icon tray on Windows (bottom right hand corner of your monitor). Or it can be run in a command prompt window and left running in the background. PhidgetWebService requires you, at minimum, to specify a password that will be used to authenticate any clients, such as your .NET application. You can change the Port, ID and Password for the PhidgetWebService the default values are Port 5001, with a Server ID of 0 and a password of "pass". The Server ID Number is used to find a unique instance of PhidgetWebService on local network without worrying about IP addresses.

Basic Example

This is a very simple example of how to access and control a PhidgetServo. It attempts to open a PhidgetServo, and if successful, will set servo motor #0 to position 90. (approximately 90 degrees)

```
var servo:PhidgetServo;
servo = new PhidgetServo();
servo.onAttach = function(phid:Phidget) {
    servo.SetMotorPosition(0,90);
    servo.Close();
};
servo.openRemoteIP("192.168.2.107",5001,-1,"pass");
```

Phidget Manager

The Phidget Manager is a useful tool for keeping track of Phidgets attached to a PhidgetWebService by firing attach and detach events for all Phidgets. This is useful if you just want to know which Phidgets are attached to the computer, or need to wait for a particular Phidget to get attached.

The Flash version of Phidget Manager does not keep a list of Phidgets, but just fires the attach and detach handlers.

In this example, we first create a Phidget Manager, then register the events handlers to print out messages about the devices, then open the Phidget Manager itself:

```
manager = new PhidgetManager();
manager.onAttach = function(phid:Phidget)
{
    trace("Attach Event: "+phid.getDeviceType()+" "+phid.getSerialNumber());
};
manager.onDetach = function(phid:Phidget)
{
    trace("Detach Event: "+phid.getDeviceType()+" "+phid.getSerialNumber());
};
manager.onError = function(desc:String, code:Number)
{
    trace("Error Event: "+desc);
};
manager.openRemoteIP("localhost",5002,-1,"pass");*/
```

Events

Setting up events in Flash is very simple. Events are defined as functions in the different classes and to receive these event, you override the function with your own function.

In this example, we override the attach event that exists for all Phidgets and the Phidget Manager, and simply print out some info about the device:

```
var manager:PhidgetManager;
manager = new PhidgetManager();
manager.onAttach = function(phid:Phidget)
{
    trace("Attach Event: "+phid.getDeviceType()+" "+phid.getSerialNumber());
};
```

Calls common to each device

The following calls and events are common to all Phidget components:

openRemoteIP (ip:String, port:Number, serial:Number, pass:String) : Number

Attempts to locate a Phidget matching your requirements on a specific PhidgetWebService on the network. To connect to a PhidgetWebService with Flash, you must know the IPAddress, or specify "localhost" for the same computer that the application is running on.

SerialNumber specifies the desired Phidget serial number, allowing the call to open a specific Phidget. Specifying -1 for the serial number will cause it to open the first available Phidget for the specific software component that you are using.

IPAddress specifies the server that the Phidget is found on. This can be the hostname, or the IP address.

Port specifies the port that the server is listening on. By default, this port is 5001, unless the server is run with a different port specified.

Password is used to authenticate with the server. A server will only accept requests from clients that send the valid password.

Close ()

Closes this Phidget object.

getSerialNumber () : Number

Returns the unique serial number of this Phidget. This number is set during manufacturing, and is unique across all Phidgets.

isAttached () : Boolean

Returns an Boolean indicating the status of the device – if it is connected.

getDeviceVersion () : Number

Returns the device version of this Phidget.

getDeviceType () : String

Returns a string describing the name of the Phidget. All PhidgetInterfaceKits will return "PhidgetInterfaceKit", PhidgetRFID returns "PhidgetRFID" and so on.

getServerAddress () : String;

Returns the IP address of the Phidget WebService hosting the device.

getServerPort () : Number;

Returns the Port of the Phidget WebService hosting the device.

getServerID () : Number;

Returns the Server ID of the Phidget WebService hosting the device.

onDetach (phid:Phidget)

A callback function that is invoked when a Phidget device associated with this software component is detached.

onAttach (phid:Phidget)

A callback function that is invoked when a Phidget device is successfully connected to. This software component can now be used to control and receive data from that phidget.

onError (desc:String, code:Number)

A callback function that is invoked when an unexpected condition occurs.

Specific devices

Each Phidget component is described along with its API.

PhidgetAccelerometer

The PhidgetAccelerometer is a component that provides a high-level programmer interface to control a PhidgetAccelerometer device connected through a USB port. With this component, the programmer can:

- Measure +-2 times Gravity (9.8 m/s^2) change per axis.
- Measure both dynamic acceleration (e.g., vibration) and static acceleration (e.g., gravity or tilt).

In addition to the common calls described above, the following calls are specific to this device:

GetNumAxis () : Number

Get the number of axes available from the PhidgetAccelerometer.

GetAcceleration (Index : Number) : Number

Gets the last acceleration value received from the PhidgetAccelerometer for a particular axis.

GetAccelerationChangeTrigger (Index : Number) : Number

Gets the amount of change that should exist between the last reported value and the current value before an OnAccelerationChange event is fired.

SetAccelerationChangeTrigger (Index : Number, NewVal : Number)

Specifies the amount of change that should exist between the last reported value and the current value before an OnAccelerationChange event is fired.

onAccelerationChange (index:Number, acceleration:Number)

A callback function that will run if the acceleration changes by more than the Acceleration trigger.

PhidgetEncoder

The PhidgetEncoder is a component that provides a high-level programmer interface to control a PhidgetEncoder device connected through a USB port. With this component, the programmer can:

- Detect changes in position of incremental and absolute encoders.
- Easily track the changes with respect to time.

In addition to the common calls described above, the following calls are specific to this device:

GetNumInputs () : Number

Returns the number of inputs on this particular Phidget device. Please see the hardware description for the details of device variations.

GetNumEncoders () : Number

Gets the number of encoders available on the Phidget.

GetEncoderPosition (Index : Number) : Number

Gets the last position value received from the encoder for the particular encoder selected.

GetInputState (Index : Number) : Boolean

Returns the state of the designated input. In the case of a switch or button which is normally open, True would correspond to when the switch is pressed.

SetEncoderPosition (Index : Number, NewVal : Number)

Specifies a new value for the current position of the encoder.

onInputChange (index:Number, newState:Boolean)

A callback function that will run if an input changes.

onPositionChange (index:Number, time:Number, encoderDisplacement:Number)

A callback function that will run if the encoder changes.

PhidgetInterfaceKit

The PhidgetInterfaceKit is a component that provides a high-level programmer interface to control a PhidgetInterfaceKit device connected through a USB port. With this component, the programmer can:

- Turn particular outputs on and off.
- Get notified of changes of state of the inputs as events.
- Configure events to fire when the analog inputs change.

The PhidgetInterfaceKit devices provide a combination of

- Digital outputs.
- Digital inputs.
- Analog inputs.

In addition to the common calls described above, the following calls are specific to this device:

GetNumOutputs () : Number

Returns the number of outputs on this particular Phidget device. Please see the hardware description for the details of device variations.

GetNumInputs () : Number

Returns the number of inputs on this particular Phidget device. Please see the hardware description for the details of device variations.

GetNumSensors () : Number

Gets the number of analog inputs available on the given PhidgetInterface Kit.

GetInputState (Index : Number) : Boolean

Returns the state of the designated input. In the case of a switch or button which is normally open. True would correspond to when the switch is pressed.

GetOutputState (Index : Number) : Boolean

Returns the state of the designated output to True or False. Depending on the type of output available the specified output goes to a high value or completes a connection. Please see the hardware description for details on different outputs.

GetSensorValue (Index : Number) : Number

Gets the last reported sensor value for the given index.

GetSensorNormalizeMinimum (Index : Number) : Number

Gets the minimum setting of a sensor's range. This is usually 0, but if one sets it to a larger value, e.g., 200, then the scale is adjusted so that the real range of 200 - 1000 are normalized to 0 - 1000. Actual readings less than this number are always reported as 0. There must be at least a difference of 1 between this property and SensorNormalizeMaximum, otherwise INVALIDARG is returned.

GetSensorNormalizeMaximum (Index : Number) : Number

Gets the maximum setting of a sensor's range. This is usually 1000, but if one sets it to a smaller value, e.g., 700, then the scale is adjusted so that the real range of 0 - 700 are normalized to 0 - 1000. Actual readings greater than this number are always reported as 1000. There must be at least a difference of 1 between this property and SensorNormalizeMinimum, otherwise an INVALIDARG error is returned.

GetSensorChangeTrigger (Index : Number) : Number

Gets the amount of change that should exist between the last reported value and the current value before an OnSensorChange event is fired.

SetOutputState (Index : Number, NewVal : Boolean)

Sets the state of the designated output to True or False. Depending on the type of output available the specified output goes to a high value or completes a connection. Please see the hardware description for details on different outputs.

SetSensorNormalizeMinimum (Index : Number, NewVal : Number)

Specifies the minimum setting of a sensor's range.

SetSensorNormalizeMaximum (Index : Number, NewVal : Number)

Specifies the maximum setting of a sensor's range.

SetSensorChangeTrigger (Index : Number, NewVal : Number)

Specifies the amount of change that should exist between the last reported value and the current value before an OnSensorChange event is fired.

onInputChange (index:Number, newState:Boolean)

A callback function that will run if an input changes.

onSensorChange (index:Number, newVal:Number)

A callback function that will run if the sensor value changes by more than the OnSensorChange trigger.

PhidgetLED

The PhidgetLED is a component that provides a high-level programmer interface to control a PhidgetLED device connected through a USB port. With this component, the programmer can:

- Control each led individually, On/Off and Brightness.

In addition to the common calls described above, the following calls are specific to this device:

GetNumLEDs () : Number

Returns the number of LED positions available in this Phidget.

GetDiscreteLED (Index : Number) : Number

Gets the brightness of an individual LED. Range of brightness is 0-100.

SetDiscreteLED (Index : Number, NewVal : Number)

Sets the brightness of an individual LED. Range of brightness is 0-100.

PhidgetMotorControl

The PhidgetMotorControl is a component that provides a high-level programmer interface to control a PhidgetMotorControl device connected through a USB port. With this component, the programmer can:

- Control direction, and start and stop DC motors.
- Control the velocity and acceleration of each DC motor.
- Read the limit switch.

In addition to the common calls described above, the following calls are specific to this device:

GetNumMotors () : Number

Returns the maximum number of motors on this particular Phidget device. This depends on the actual hardware. Please see the hardware description for the details of device variations.

Note: that there is no way of programmatically determining how many motors are actually plugged into the hardware.

GetNumInputs () : Number

Returns the number of inputs on this particular Phidget device. Please see the hardware description for the details of device variations.

GetInputState (Index : Number) : Boolean

Returns the state of the designated input. In the case of a switch or button which is normally open, True would correspond to when the switch is pressed.

GetAcceleration (Index : Number) : Number

desc

GetMotorSpeed (Index : Number) : Number

desc

SetAcceleration (Index : Number, NewVal : Number)

desc

SetMotorSpeed (Index : Number, NewVal : Number)

desc

onInputChange (index:Number, newState:Boolean)

A callback function that will run if an input changes.

onMotorChange (index:Number, newVal:Number)

A callback function that will run when the motor speed changes.

PhidgetPHSensor

The PhidgetPHSensor is a component that provides a high-level programmer interface to control a PhidgetPHSensor device connected through a USB port. With this component, the programmer can:

- Read the ph of a solution.

In addition to the common calls described above, the following calls are specific to this device:

GetPH () : Number

Returns the current ph.

GetPHChangeTrigger () : Number

Gets the amount of change that should exist between the last reported value and the current value before an OnPHChange event is fired.

SetPHChangeTrigger (NewVal : Number)

Specifies the amount of change that should exist between the last reported value and the current value before an OnPHChange event is fired.

onPHChange (ph:Number)

A callback function that will run if the PH changes by more than the PH trigger.

PhidgetRFID

The PhidgetRFID is a component that provides a high-level programmer interface to control a PhidgetRFID device connected through a USB port. With this component, the programmer can:

- Read Radio Frequency Identification tags.

Radio Frequency Identification or RFID, is a non-contact identification technology which uses a reader to read data stored on low cost tags. The particular instance of the technology we use stores a 40-bit number on the tag. Every tag that is purchased from Phidgets Inc. is guaranteed unique.

When a RFID tag is read, the component returns the unique number contained in the RFID tag.

In addition to the common calls described above, the following calls are specific to this device:

GetNumOutputs () : Number

Returns the number of outputs on this particular Phidget device. Please see the hardware description for the details of device variations.

GetOutputState (Index : Number) : Boolean

Returns the state of the designated output to True or False. Depending on the type of output available the specified output goes to a high value or completes a connection. Please see the hardware description for details on different outputs.

SetOutputState (Index : Number, NewVal : Boolean)

Sets the state of the designated output to True or False. Depending on the type of output available the specified output goes to a high value or completes a connection. Please see the hardware description for details on different outputs.

onTag (tag:String)

A callback function that will run when a Tag is read.

PhidgetServo

The PhidgetServo is a component that provides a high-level programmer interface to control a Phidget-Servo device connected through a USB port. With this component, the programmer can:

- Set the desired position for a servo motor, ranging from 0 to 180 degrees.

In addition to the common calls described above, the following calls are specific to this device:

GetMotorPosition (Index : Number) : Number

Gets the desired servo motor position for a particular servo motor.

SetMotorPosition (Index : Number, NewVal : Double)

Sets the desired servo motor position for a particular servo motor. This value may range from -23 to 231, corresponding to time width of the control pulse, the angle that the Servo motor moves to depends on the characteristic of individual motors. Please read the PhidgetServo documentation to understand what behaviour you can expect from your motors.

GetNumMotors () : Number

Returns the maximum number of motors on this particular Phidget device. This depends on the actual hardware. Please see the hardware description for the details of device variations.

Note that there is no way of programmatically determining how many motors are actually plugged into the hardware.

PhidgetTemperatureSensor

The PhidgetTemperatureSensor is a component that provides a high-level programmer interface to control a PhidgetTemperatureSensor device connected through a USB port. With this component, the programmer can:

- Read the temperature of Thermocouple device.
- Read cold junction temperature.
- Get notification of temperature change.
- Use metric or imperial units.

In addition to the common calls described above, the following calls are specific to this device:

GetNumTemperatureInputs () : Number

Returns the integer value of the number of thermocouple inputs.

GetTemperature (Index : Number) : Number

Returns the current temperature in Celsius or Fahrenheit (depending on UseImperial property). Index = 0 returns the temperature of the cold junction. Index = 1 returns the temperature of the thermocouple.

GetTemperatureChangeTrigger (Index : Number) : Number

Gets the amount of change that should exist between the last reported value and the current value before an OnTemperatureChange event is fired.

SetTemperatureChangeTrigger (Index : Number, NewVal : Number)

Specifies the amount of change that should exist between the last reported value and the current value before an OnTemperatureChange event is fired.

SetUseImperial (NewVal : Boolean)

Specifies the state indicating if Metric (SI units) or Imperial (USA, Myanmar, Liberia) units are being used.

onTemperatureChange (index:Number, temperature:Number)

A callback function that will run if the Temperature changes by more than the Temperature trigger.

PhidgetTextLCD

The PhidgetTextLCD is a component that provides a high-level programmer interface to control a PhidgetTextLCD device connected through a USB port. With this component, the programmer can:

- Display text on a PhidgetTextLCD module.

In addition to the common calls described above, the following calls are specific to this device:

GetNumRows () : Number

Returns number of rows of text that may be presented on the display.

GetNumColumns () : Number

Returns number of columns of text that may be used on the display.

GetBacklight () : Boolean

Determines if the backlight for this LCD is on or off.

GetCursorOn () : Boolean

Determines if the cursor is on or off.

GetCursorBlink () : Boolean

Determines if the cursor's blinking is on or off.

SetBacklight (NewVal : Boolean)

Sets the backlight for this LCD on or off.

SetCursorOn (NewVal : Boolean)

Sets the cursor on or off. This cursor is displayed at the last location that was changed.

SetCursorBlink (NewVal : Boolean)

Sets the cursor's blinking on or off.

SetDisplayString (Row : Number, DisplayString : String)

Sets the text to display on a particular row of the display. The text will be clipped at the right edge of the display.

PhidgetTextLED

The PhidgetTextLED is a component that provides a high-level programmer interface to control a PhidgetTextLED device connected through a USB port. With this component, the programmer can:

- Display text and numbers on segment type LED modules.
- Brightness can be controlled for the entire display.

In the case of 7-segment LED characters numbers are displayed easily and text can be displayed with some restrictions.

In addition to the common calls described above, the following calls are specific to this device:

GetNumRows () : Number

Returns number of rows of text that may be presented on the display.

GetNumColumns () : Number

Returns number of columns of text that may be used on the display.

GetBrightness () : Number

Returns the brightness value of the LED display.

SetBrightness (NewVal : Number)

Sets the brightness of the LED display. Varying this property will control the brightness uniformly across all digits in the display.

SetDisplayString (Row : Number, DisplayString : String)

Sets the text to display on a particular row of the display. Will clip the text at the right edge of the display.

PhidgetWeightSensor

The PhidgetWeightSensor is a component that provides a high-level programmer interface to control a PhidgetWeightSensor device connected through a USB port. With this component, the programmer can:

- Read the weight of an item or person on the weight scale.

In addition to the common calls described above, the following calls are specific to this device:

GetWeight () : Number

Returns the current weight in Kilograms or Pounds (depending on UseImperial property).

GetWeightChangeTrigger () : Number

Gets the amount of change that should exist between the last reported value and the current value before an OnWeightChange event is fired.

SetWeightChangeTrigger (NewVal : Number)

Specifies the amount of change that should exist between the last reported value and the current value before an OnWeightChange event is fired.

SetUseImperial (NewVal : Boolean)

Specifies the state indicating if Metric (SI units) or Imperial (USA, Myanmar, Liberia) units are being used.

onWeightChange (weight:Number)

A callback function that will run if the Weight changes by more than the Weight trigger.