PythonInterface User Guide.

Python Related Information

I don’t intend to cover Python installation, please visit the Python web site for details.

http://www.python.org/

Before you start to use the Python Interface plugin make sure that your Python installation is working properly.

Installation

Get the latest plugin from my website.

http://www.xpluginsdk.org/python_interface.htm

Make sure that you get the version that matches the version of Python that you have installed, this is very important.

If your OS has python 2.6 installed then make sure you use a PythonInterface plugin that has a 26 suffix.
e.g. PythonInterfaceXXX26.xpl, were XXX is Win, Lin or Mac depending on the OS that you are using.

Extract the plugin and the INI file from the zip file.
Copy the PythonInterfaceXXX26.xpl and the PythonInterface.ini files to the plugins folder.
e.g. copy to “D:\X-Plane 9.00\Resources\plugins”

This could be C:\ or any other drive letter depending where you have installed XPlane.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

PythonInterface.ini file

The PythonInterface.ini is only recognized with the new version of the PythonInterface plugin (currently 2.66.01).
It is included in the PythonInterfaceXXX26.zip for version 2.66.01.
Where XXX is Win, Mac or Lin depending on the OS you are using.

The settings in the file are shown below, these are the default if the file is not present.

```
[DEBUG]
DebugCallbacks = 0
CheckCallbacks = 0
```

Setting DebugCallbacks to 1 (one) will write debug info on callbacks to the PythonInterfaceLog.txt file.
This is very useful for checking if you are de-registering callbacks.
Setting CheckCallbacks to 1 (one) will mean that the plugin will check that all callbacks that have been registered have been de-registered on exit.
The results of this check are written to the PythonInterfaceLog.txt file.

Text files that are created in the PythonScripts Directory

PythonInterfaceLog.txt file.

This is so I can diagnose any problems.
It captures any errors that occurs within the plugin and will also show script error info.
PythonInterfaceOutput.txt file.

This captures any print statements.
This can be used by a user who wants to keep any of their output data.
There is a check box on the Control Panel to enable or disable this output.
The check box state is not persistent so it will always set on after Xplane startup.
This is so that a user can capture print output during startup.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

Examples

Download the Example Scripts from my website.

http://www.xpluginsdk.org/downloads/PythonScripts.zip

Unzip these into the above plugins directory so that PythonScripts is a sub directory of the plugins directory.

e.g. you should have this hierarchy.

D:\X-Plane 9.00\Resources\plugins
D:\X-Plane 9.00\Resources\plugins\PythonScripts
D:\X-Plane 9.00\Resources\plugins\PythonScripts\AdvancedSDKExamples
D:\X-Plane 9.00\Resources\plugins\PythonScripts\SDKExamples

Official Plugin SDK Documentation

Also take some time to read the docs on our plugin SDK website.

http://www.xsquawkbox.net/xpsdk/phpwiki/index.php?Documentation

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

Testing the Installation

The best way to test the installation is to use the “PI_HelloWorld1.py” script which can be found in the
 “D:\X-Plane 8.20\Resources\plugins\PythonScripts\SDKExamples” directory.

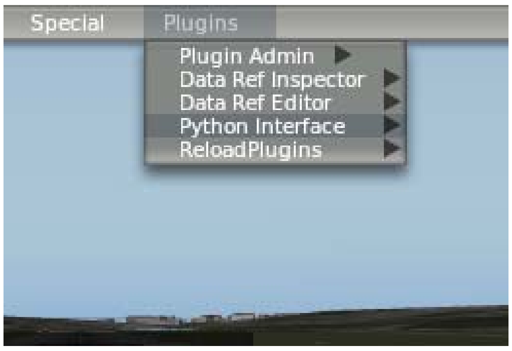
Copy “PI_HelloWorld1.py” to the “plugins\PythonScript” directory on the version of Xplane that you are using.

e.g.

D:\X-Plane 9.00\Resources\plugins\PythonScript

Start Xplane and you should see a translucent window with the text “Hello world 1” in it. Left click with the mouse and it should change to “I'm a plugin 1”.

If this does not appear then select the Plugins menu and select Python Interface, Control Panel.



Check for error messages and make a note of them.
If there are none then we need to check the Xplane log file.

Exit Xplane and open the Log.txt file.
In the file, look for the “Using path below for script path”.
Make sure it matches the PythonScript path.

e.g. You will always find something like this in the file.

Using path below for script path
D:\X-Plane 9.00\Resources\plugins\PythonScript

Also check for any other errors from the PythonInterface plugin.

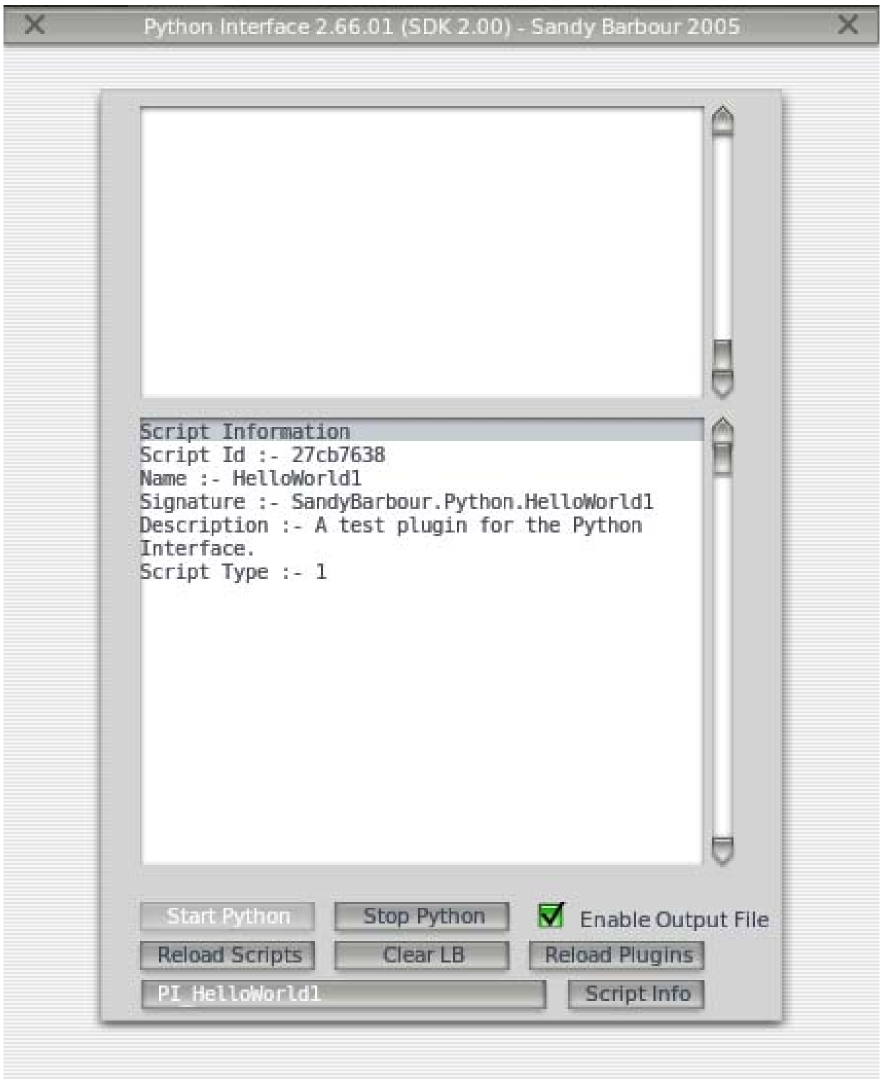
If everything is working and you can see the script, I will explain the purpose of the PythonInterface menu options.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

PythonInterface Menu

Control Panel



```
Script Information
Script Id :- 27cb7638
Name :- HelloWorld1
Signature :- SandyBarbour.Python.HelloWorld1
Description :- A test plugin for the Python
Interface.
Script Type :- 1
```

The control panel is used for the following purposes.

To display error messages and other messages.
To Reload Scripts after modifying a script, great for testing ideas.
To get information on a script, select the script in the Popup and then left click on the Script Info button.

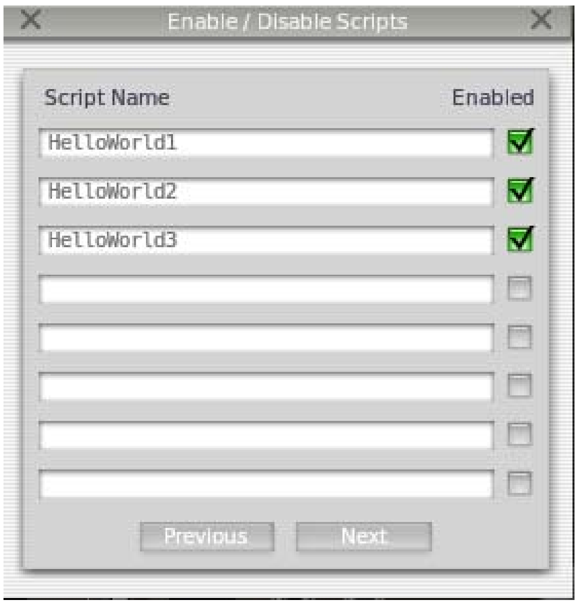
To Reload Plugins if things start to go wrong.

If you are using menus then Reload Scripts will add another menu.
In this case use Reload Plugins to reset the menus.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

Enable/Disable Scripts Panel



Left click on the Enabled checkbox to enable/disable a script.
A green check mark means enabled, no tick mark means disabled.

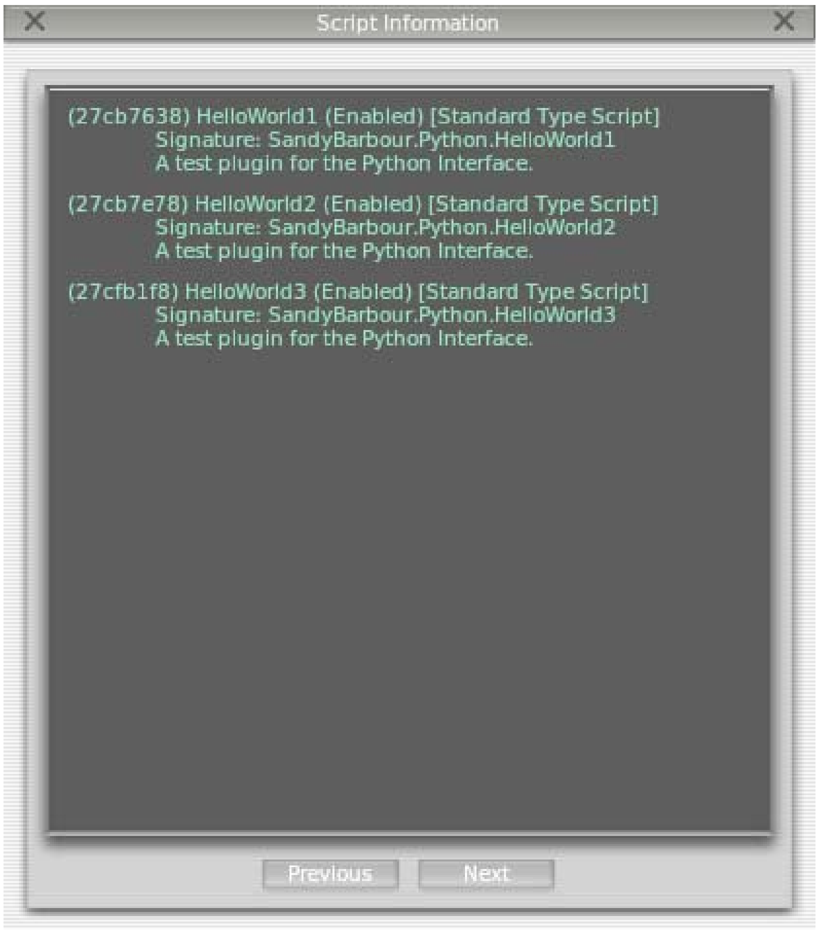
If there are more than 8 scripts the Next button will be highlighted.
Left click on the Next button to access more scripts.

The Previous button will be disabled when it is on the first page.
The Next button will be disabled when it is on the last page.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

Script Information Panel



```
(27cb7638) HelloWorld1 (Enabled) [Standard Type Script]
        Signature: SandyBarbour.Python.HelloWorld1
        A test plugin for the Python Interface.

(27cb7e78) HelloWorld2 (Enabled) [Standard Type Script]
        Signature: SandyBarbour.Python.HelloWorld2
        A test plugin for the Python Interface.

(27cfb1f8) HelloWorld3 (Enabled) [Standard Type Script]
        Signature: SandyBarbour.Python.HelloWorld3
        A test plugin for the Python Interface.
```

This panel displays information for each script.

If there are more than 8 scripts the Next button will be highlighted.
Left click on the Next button to access more scripts.

The Previous button will be disabled when it is on the first page.
The Next button will be disabled when it is on the last page.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

PythonInterface User Guide.

The following functions can be used to display your own messages from within a script.

SandyBarbourDisplay()
This will send text to the top widget listbox.

SandyBarbourClearDisplay()
This will clear the text in the top widget listbox.

SandyBarbourPrint()
This will send text to the bottom widget listbox.

SandyBarbourClearPrint()
This will clear the text in the bottom widget listbox.

The "SandyBarbourDisplay()" and "SandyBarbourClearDisplay()" are useful for displaying changing data from Xplane.

e.g.

```
def FlightLoopCallback(self, elapsedMe, elapsedSim, counter, refcon):
        SandyBarbourClearDisplay()
        SandyBarbourDisplay("Throttle 1 :- " + str(self.Throttles[0]))
        SandyBarbourDisplay("Throttle 2 :- " + str(self.Throttles[1]))
        SandyBarbourDisplay("Throttle 3 :- " + str(self.Throttles[2]))
        SandyBarbourDisplay("Throttle 4 :- " + str(self.Throttles[3]))
        return 0.1
```

This will show a constant update of the throttle data in the top listbox.

The "SandyBarbourPrint()" and "SandyBarbourClearPrint()" can be used to log messages in the bottom listbox.

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

Debugging Scripts

Use the bottom listbox to debug your script.
Any errors will appear there.

On a new script open the control panel after Xplane has started.
The bottom list box will show any errors that occurred in XPluginStart.
It will give you the line number to help you located the error.

Use Reload Scripts after you have edited your script.

If you still get errors, try a Reload Plugins.

Occasionally you will need to restart Xplane.

PythonInterface User Guide.

Starting a Script from Scratch.

A script consists of the following.

```
class PythonInterface:
        def XPluginStart(self):
                self.Name = "Template"
                self.Sig =  "SandyBarbour.Python.Template"
                self.Desc = "A test script for the Python Interface."
                return self.Name, self.Sig, self.Desc

        def XPluginStop(self):
                pass

        def XPluginEnable(self):
                return 1

        def XPluginDisable(self):
                pass

        def XPluginReceiveMessage(self, inFromWho, inMessage, inParam):
                pass
```

This the minimum required for a script to load, it will be loaded but won't be entertaining.
It is the same requirements as a plugin.

We will now add more functionality to turn it into a HelloWorld type script.

The first thing to add is the modules that contain the functions that we want to use.
At the start of the script we add these.

```
from XPLMDisplay import *
```

Needed for these functions.
XPLMCreateWindow()
XPLMDestroyWindow()
XPLMGetWindowGeometry()

```
from XPLMGraphics import *
```

Needed for these functions.
XPLMDrawTranslucentDarkBox()
XPLMDrawString()

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011

Next we add code to "XPluginStart" to create the window.

This is used to save the state of the mouse click.

self.Clicked = 0

Because we want to use it in other class functions we use self which makes it a class property.

Next we create our window to display the text.

self.DrawWindowCB = self.DrawWindowCallback
self.KeyCB = self.KeyCallback
self.MouseClickCB = self.MouseClickCallback

You need to create a copy of the callback functions.
Using the callback names alone does not work.

The following function creates our window and registers the relevant callbacks.

self.WindowId = XPLMCreateWindow(self, 50, 600, 300, 400, 1, self.DrawWindowCB, self.KeyCB, self.MouseClickCB, 0)

Because we want to destroy the window in the XPluginStop callback we use self.WindowId.

This is used by the the XPLMDestroyWindow() function as follows.

XPLMDestroyWindow(self, self.WindowId)

We don't have any code for "XPluginEnable", "XPluginDisable" and "XPluginReceiveMessage" so we leave them as they are.

The next thing to do is add the Draw Window Callback function

```
def DrawWindowCallback(self, inWindowID, inRefcon):
	pass
```

We then add this code to the function.

```
lLeft = []; lTop = [];  lRight = []; lBottom = []
XPLMGetWindowGeometry(inWindowID, lLeft, lTop, lRight, lBottom)
```

We use empty lists for the parameters of "XPLMGetWindowGeometry()" as it will fill them with the values that it gets from the SDK function of the same name.

We now need to copy these into variables that we will use later on.
There is only one item in the list so we use list element zero in each case.

```
left = int(lLeft[0]); top = int(lTop[0]); right = int(lRight[0]); bottom = int(lBottom[0])
```

We pass these variables into this function to draw our Translucent window.
This allows Xplane objects to be seen through this window.

```
gResult = XPLMDrawTranslucentDarkBox(left, top, right, bottom)
```

We set the text colour to white, change to suit your own preferences.

```
colour = 1.0, 1.0, 1.0
```

We now test the class property (variable) that we created in XPluginStart.

```
if self.Clicked :
	Desc = "I'm a plugin 1"
else:
	Desc = "Hello World 1"
```

If it is true (mouse button pressed) we will display "I'm a plugin 1", if it is not true (no button pressed) we will display "Hello World 1".

These are saved into a variable for use in the next function.

```
gResult = XPLMDrawString(colour, left + 5, top - 20, Desc, 0, xplmFont_Basic)
```

This will display the text in the selected colour and position.

We now add the Mouse Callback function

```
def MouseClickCallback(self, inWindowID, x, y, inMouse, inRefcon):
        pass
```

We then add the following code to it.

```
if ((inMouse == xplm_MouseDown) or (inMouse == xplm_MouseUp)):
        self.Clicked = 1 - self.Clicked
```

We use the function parameter *inMouse* to get the mouse button state.
We are only interested in the mouse button down and up.
This code will ensure that the self.Clicked property will toggle between the two states, true and false when the mouse button is clicked.

We add the Key Callback function but there is no code to put in it.

```
def KeyCallback(self, inWindowID, inKey, inFlags, inVirtualKey, inRefcon,
losingFocus):
        pass
```

However it is required even though we are not using it.

The finished script is shown on the next page.

This is obviously a very simple example, but shows you the basics in getting a script up and running.

The rest comes with reading the SDK docs and learning all the functions that are available.

I have wrapped all the functionality of the SDK so there should be no limits, in theory, of what can be done.

It is also a very good idea to study all the SDK examples that I ported to python.
Start with the standard examples and then work up to the advanced examples.

PythonInterface User Guide.

```
from XPLMDisplay import *
from XPLMGraphics import *

class PythonInterface:
	def XPluginStart(self):
		self.Name = "Template"
		self.Sig =  "SandyBarbour.Python.Template"
		self.Desc = "A test script for the Python Interface."

		self.Clicked = 0
		self.DrawWindowCB = self.DrawWindowCallback
		self.KeyCB = self.KeyCallback
		self.MouseClickCB = self.MouseClickCallback
		self.WindowId = XPLMCreateWindow(self, 50, 600, 300, 400, 1, self.DrawWindowCB,
self.KeyCB, self.MouseClickCB, 0)

		return self.Name, self.Sig, self.Desc

	def XPluginStop(self):
		XPLMDestroyWindow(self, self.WindowId)
		pass

	def XPluginEnable(self):
		return 1

	def XPluginDisable(self):
		pass

	def XPluginReceiveMessage(self, inFromWho, inMessage, inParam):
		pass

	def DrawWindowCallback(self, inWindowID, inRefcon):
		lLeft = [];		lTop = []; lRight = [];	lBottom = []
		XPLMGetWindowGeometry(inWindowID, lLeft, lTop, lRight, lBottom)
		left = int(lLeft[0]); top = int(lTop[0]); right = int(lRight[0]); bottom = int(lBottom[0])

		gResult = XPLMDrawTranslucentDarkBox(left, top, right, bottom)
		colour = 1.0, 1.0, 1.0

		if self.Clicked :
			Desc = "I'm a plugin 1"
		else:
			Desc = "Hello World 1"

		gResult = XPLMDrawString(colour, left + 5, top - 20, Desc, 0, xplmFont_Basic)
		pass

	def KeyCallback(self, inWindowID, inKey, inFlags, inVirtualKey, inRefcon, losingFocus):
		pass

	def MouseClickCallback(self, inWindowID, x, y, inMouse, inRefcon):
		if ((inMouse == xplm_MouseDown) or (inMouse == xplm_MouseUp)):
			self.Clicked = 1 - self.Clicked

		return 1
```

Created by Sandy Barbour – 9th November, 2005
Updated by Sandy Barbour -Sunday, 06 March 2011