
Python

Release

Jun 27, 2017

Contents

1	Controlling your living room with EventGhost	3
1.1	Introduction	3
1.2	IR PC Control	3
1.3	RF X10 PC Control	4
1.4	IR Device Control	4
1.5	Programming	4
1.6	Results	11
1.7	Conclusion	11
2	Short Manual	13
2.1	Main Window	13
2.2	Logger Pane	14
2.3	Configuration Pane	14
2.4	Adding Plugins	15
2.5	Assigning Events to Macros	15
3	Features	17
4	Supported Remote Receivers	19
4.1	Already supported devices	19
4.2	Not directly supported devices	20
5	List of Plugins	21
5.1	Essential (always loaded)	21
5.2	Remote Receiver	21
5.3	Program Control	22
5.4	External Hardware Equipment	23
5.5	Other	24
6	Scripting with Python	27
6.1	Introduction	27
6.2	The one and all ‘eg’ object	27
7	FAQ	31
7.1	Q: Isn’t the installer huge for such a tool?	31
7.2	Q: Can I use an IrDA dongle to control my PC with a remote?	31
7.3	Q: How can I install EventGhost on a Windows 2000 machine?	32

8	Command Line Options	33
9	Contributing	35
10	X10 Remotes	37
11	Indices and tables	39

Welcome! This is the user documentation for EventGhost , last updated Jun 27, 2017.

Controlling your living room with EventGhost

This article is a thank-worthy contribution by Benjamin Webb.

Introduction

Yes, I've been there. You have your HTPC all set up to go. Then you realize that you have no way to seamlessly use your HTPC with your current living room set up. You may be sitting there with a wireless keyboard and mouse controlling it or you may be plopped down in front of your PC pressing return to select a different video to be displayed. There is hope for you yet. There are many solutions today for giving you control of both your PC and the remaining elements of your living room. Currently, most universal IR remotes are up to the task of linking together most things in your living room except your PC. Many places offer niche products that are set up to control certain applications on a computer or record and emit a limited number of IR codes. I have long since searched for the ultimate of HTPC building: pressing a button and having it turn the TV on, and then go on to control your PC. There are many schools of thought as to how this should occur. This article teaches you one that involves an innovative, free program called EventGhost for Windows.

IR PC Control

I love basic IR PC control. It is inexpensive and, thanks to EventGhost, easy to set up. The downside is pretty obvious. You need line of sight. You cannot turn up the volume when someone wanders in front of the TV as a result. People in general are used to the basic concept of IR, just point and click. It is inexpensive since you can use whatever remote you have laying around. One piece of advice is to try to get a remote that sends the same signal every time you click the button. Some remotes have a rotation of signals that they send to the device they control. This can cause problems when programming such a device. You can still program remotes like this but it's annoying to put in all the other IR codes as actions that trigger events. There may even be some company out there that created a remote with a constantly changing IR signal that is interpreted by an algorithm on the device (I haven't found one yet but they may exist). The reason for not keeping IR signal constant is to prevent universal remote from doing their job and forcing the user to use the device creator's remote.

RF X10 PC Control

The program comes with an X10 remote plugin. This plugin, in theory, allows you to use any X10 compatible remote. I will test a Snapstream Firefly X10 remote in this article. I have tested this remote before and it will travel 35 feet through just about any wall, excluding the cinder block walls at my dorm. The software for this remote was very resource heavy and the main reason I was searching for an alternative program.

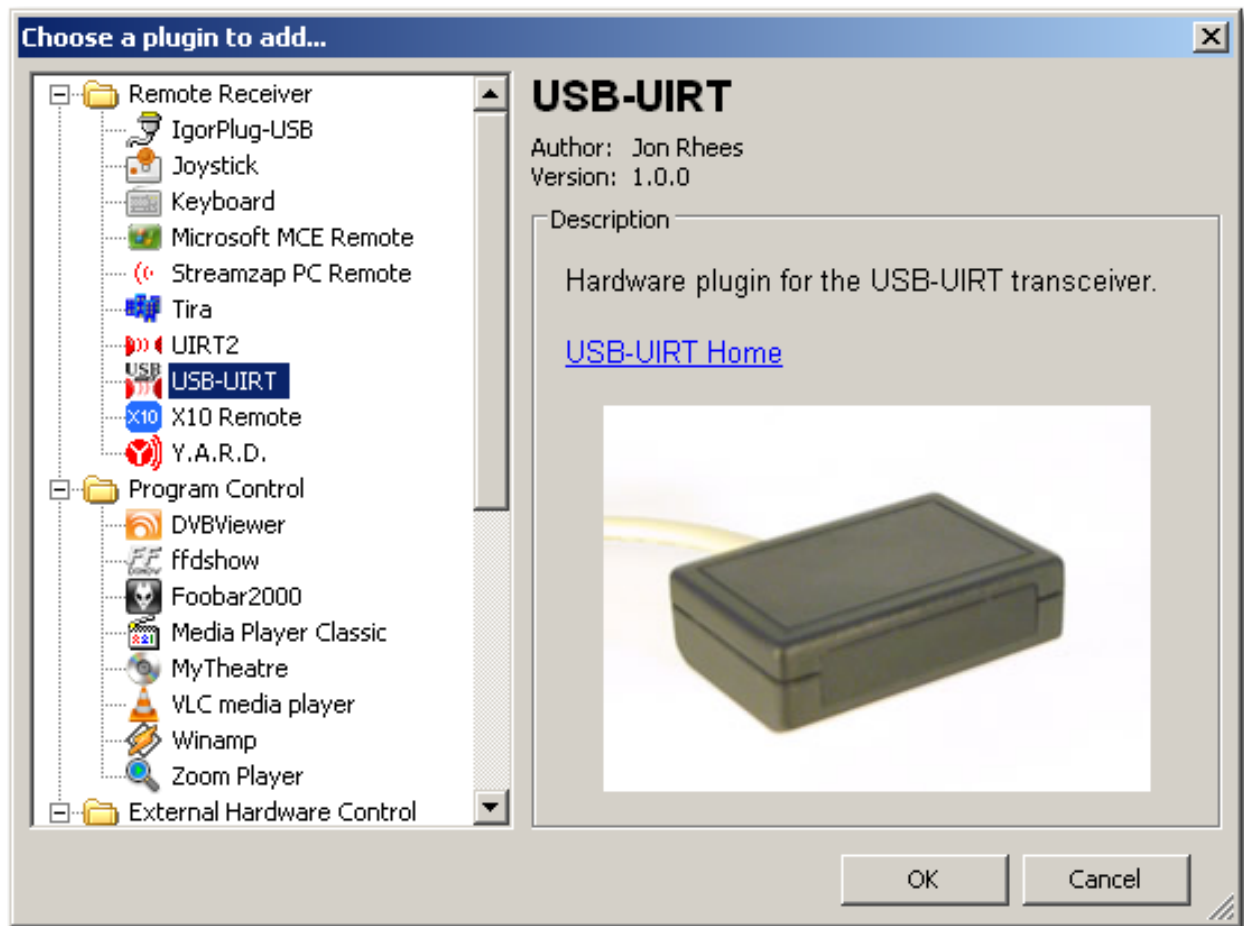
IR Device Control

EventGhost can be used with devices such as the USB-UIRT to transmit IR signals to devices you use with or without line of sight or simply use devices your remote is not programmed for. The USB-UIRT can be connected to objects placed away from line of sight. You just attach IR emitters to reach up to 300' around your house. Without the emitters the device needs to be placed in front of the device it will control. In this review, I will test two IR emitters attached with a stereo splitter. The IR emitters I will test have a reach of 10 feet.

Programming

Basics

EventGhost is a very interesting program. You install plugins that enable different events to be displayed on the logger. Simply click the *Add Plugin* button to gain access to the plugin listing screen.

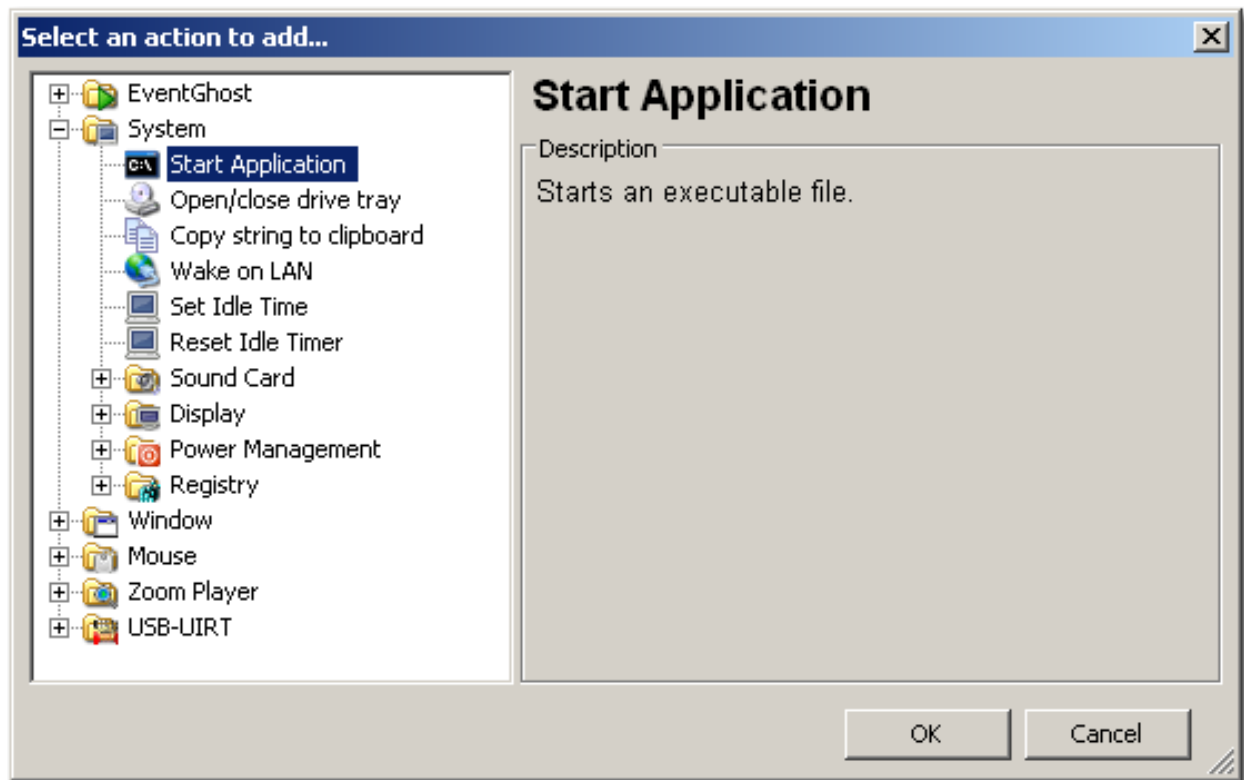


As you can see, the plugins give this program capabilities far beyond what this article covers but before you tackle network control or speech control you must learn the basics.

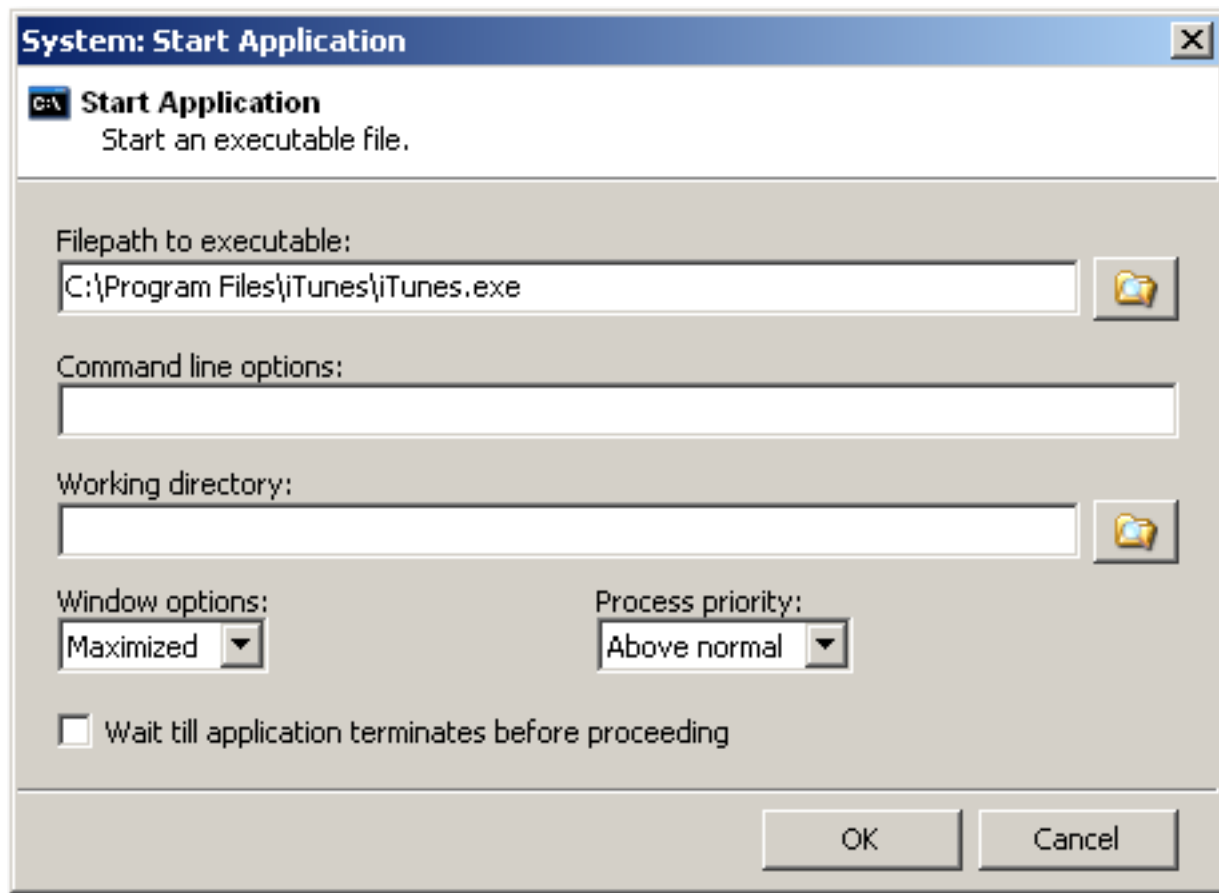
An event triggers a macro and a macro consists of one or more actions. For example, I set up the disc button on my remote to launch iTunes. First, I installed the USB-UIRT plugin. Then, when I pointed the remote toward the USB-UIRT and pressed the disc button on the remote the IR code was displayed in the logger.

	Description
★	USB_UIRT.804A5000E1F0
★	USB_UIRT.ButtonReleased

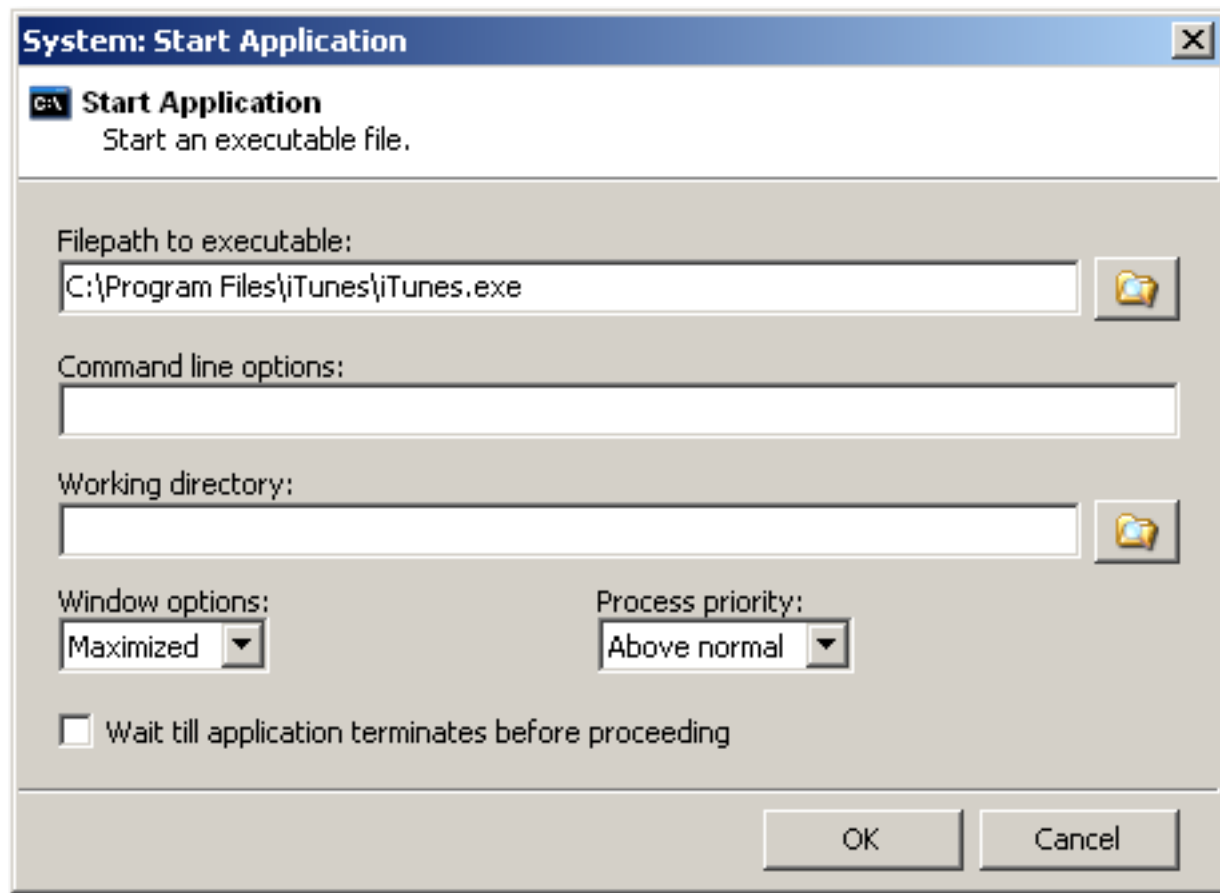
Make a macro by right clicking on the configuration panel (the group of stuff on the right) and select *Add Macro*. You will now immediately get a dialog where you can choose your first action.



After you have opened the action selection menu you just select the action of your choice. In my case, I wanted to launch iTunes, so I opened up the system folder and selected Run Application. EventGhost will now ask for additional properties of the action.



From there navigate the file path to the program you want to open. In my case, the path was `C:\Program Files\iTunes\iTunes.exe`. After you have pressed the *OK* button, you will see your newly added macro with the action in your configuration. Then drag and drop the IR event code you want to use from the logger to the macro you just created.



EventGhost names the macro automatically but you can also rename the item (for your sanity be sure the name makes sense to you)

Program Control

Now that you have the program open, there are multiple programming choices you can make. To simplify things, first make a folder in which you store all of the instructions for controlling the program. You name the folder, which I called iTunes, then you proceed by creating a macro for each action(s) you want to perform.

Enabling Items Exclusively

The Enable Item Exclusively action enables the specified macro or folder and disables all other macros and folders at the same folder level. The action can be added by right clicking and select *Add Action*. You then proceed to open up the EventGhost folder. You have to select the folder/macro to make active after adding the exclusive.

The event you want to activate your folder with is a matter of choice. I installed the Task Create/Switch Events plugin to cause the folder to be enabled when iTunes is active. If you choose to activate the folder when you launch iTunes from the disc button this will also work fine. You can activate other folders when you close programs or press a different button also. The choices are all up to you.

The program comes with examples from WinAmp and ZoomPlayer. If these are your media programs of choice then all you need to do is replace the events put there by default with the ones from your IR remote. I understand that this is a brief explanation but it should enable you to be able to perform most basic keyboard shortcuts with the program

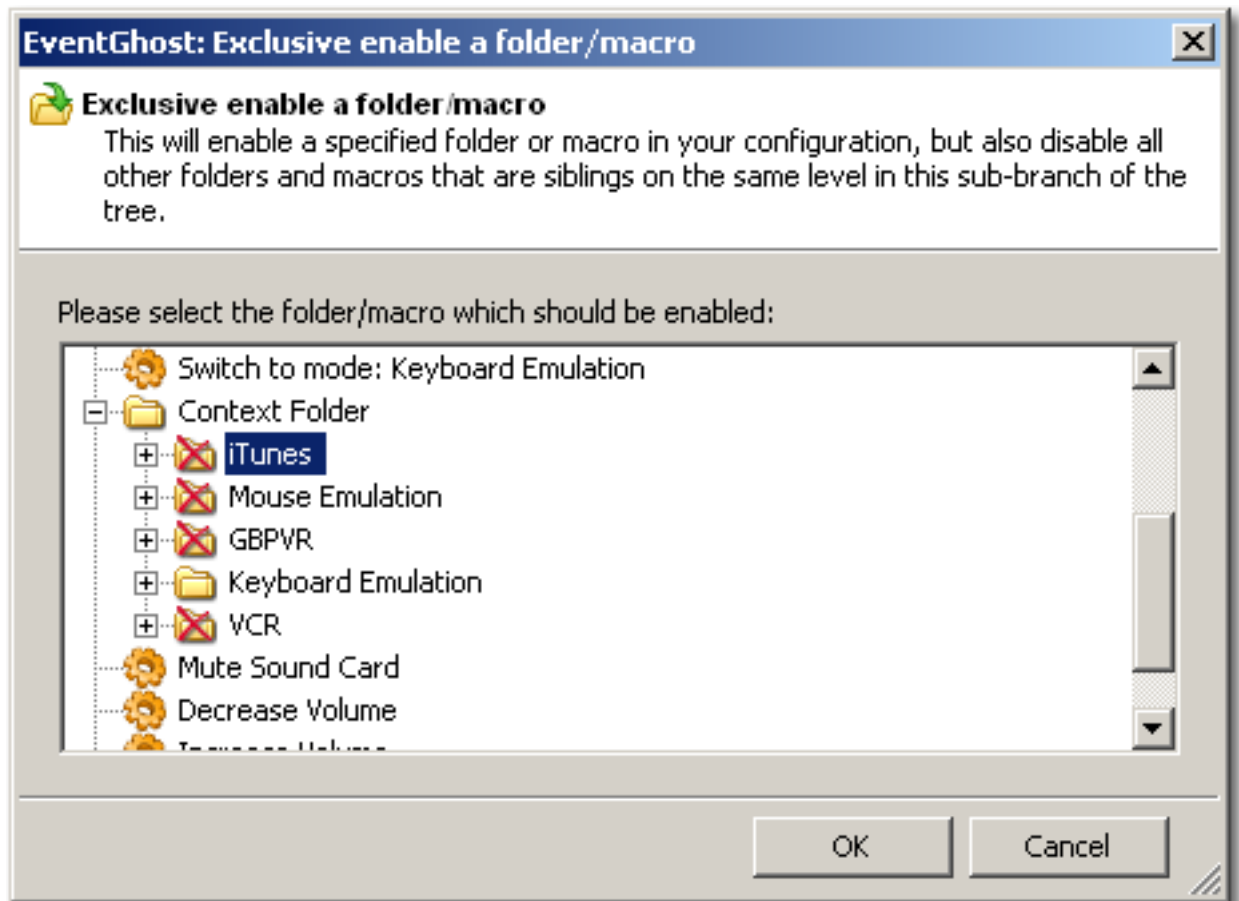
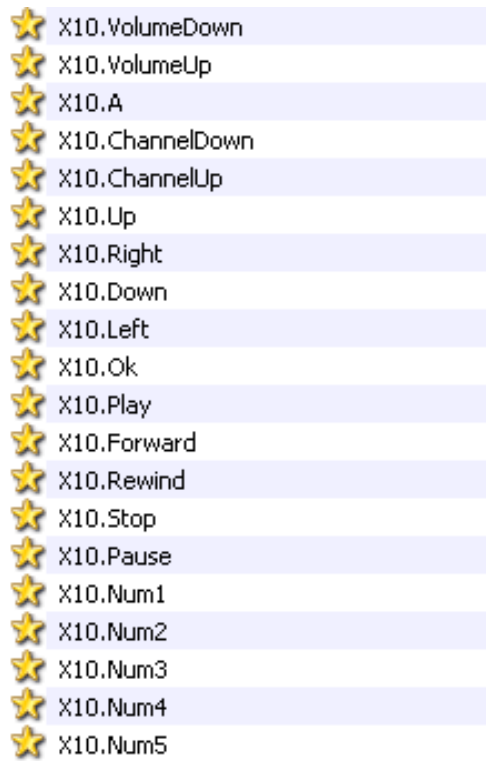
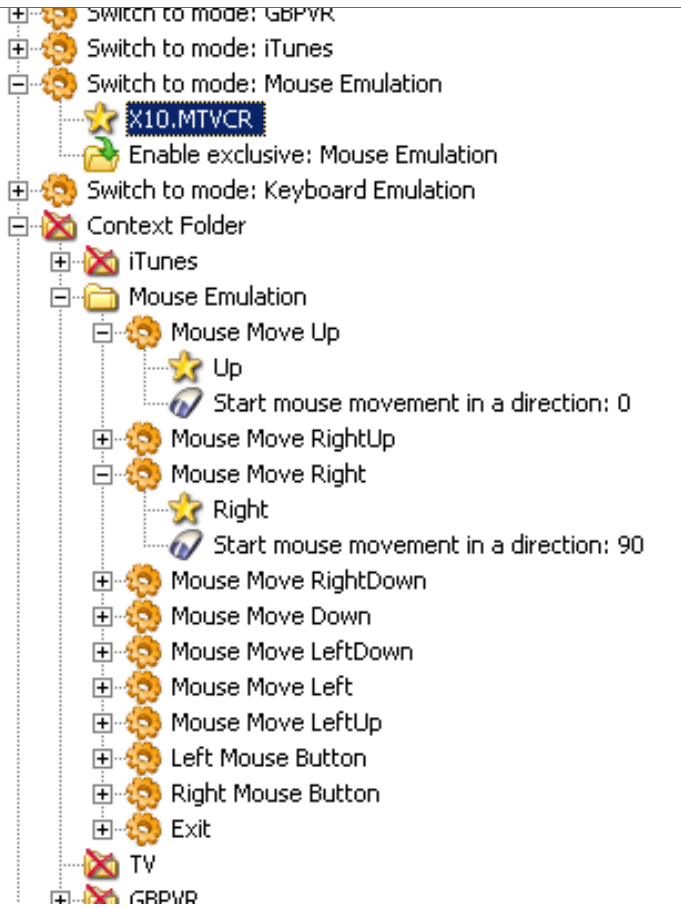


Fig. 1.1: Just select the folder that you want to make active as a result of the event in the macro.

of your choice. EventGhost is mostly drag and drop or copy and paste. EventGhost also includes an excellent mouse emulator that you can tweak to your satisfaction.

X10 Programming

IR remote and X10 remote instructions remain basically the same. The events are simply listed on the logger as with the IR. The mouse functionality was easily implemented by simply putting the X10 event X10.MTVCR in the “switch to mode: Mouse Emulation” macro. I then programed in the directional keys. I was not able to completely fine tune the X10.xml to be activated by the opening of iTunes or GBPVR due to the death of my remote. Here is a perfect start for those willing to tweak some more.

This is a sample of some X10 events on the logger:	This is what I did to enable the mouse on the Firefly:
 <ul style="list-style-type: none"> ★ X10.VolumeDown ★ X10.VolumeUp ★ X10.A ★ X10.ChannelDown ★ X10.ChannelUp ★ X10.Up ★ X10.Right ★ X10.Down ★ X10.Left ★ X10.Ok ★ X10.Play ★ X10.Forward ★ X10.Rewind ★ X10.Stop ★ X10.Pause ★ X10.Num1 ★ X10.Num2 ★ X10.Num3 ★ X10.Num4 ★ X10.Num5 	 <ul style="list-style-type: none"> Switch to mode: GBPVR Switch to mode: iTunes Switch to mode: Mouse Emulation <ul style="list-style-type: none"> X10.MTVCR Enable exclusive: Mouse Emulation Switch to mode: Keyboard Emulation Context Folder <ul style="list-style-type: none"> iTunes Mouse Emulation <ul style="list-style-type: none"> Mouse Move Up <ul style="list-style-type: none"> Up Start mouse movement in a direction: 0 Mouse Move RightUp <ul style="list-style-type: none"> Mouse Move Right <ul style="list-style-type: none"> Right Start mouse movement in a direction: 90 Mouse Move RightDown Mouse Move Down Mouse Move LeftDown Mouse Move Left <ul style="list-style-type: none"> Mouse Move LeftUp Left Mouse Button Right Mouse Button Exit TV GBPVR

IR Transmission Programming

This is basically more of the same. You simply create a macro with an event that triggers the IR transmitting action. This is available if you expand the USB-UIRT folder in the add actions menu.

After the action has been added you simply click on Transmit IR icon and select learn an IR code. Make sure that you position the remote closer than half an inch from the USB-UIRT to learn the IR code. I found the code was not learned correctly otherwise.

If you are looking to control channel changing for your tuner, currently you have to rely on the software that does the recording to change the channel. With the USB-UIRT this is not usually a problem since most HTPC recording

programs provide their own interfaces for programming this in. I'm sure with more advanced plugins and creative programming this may be possible.

Results

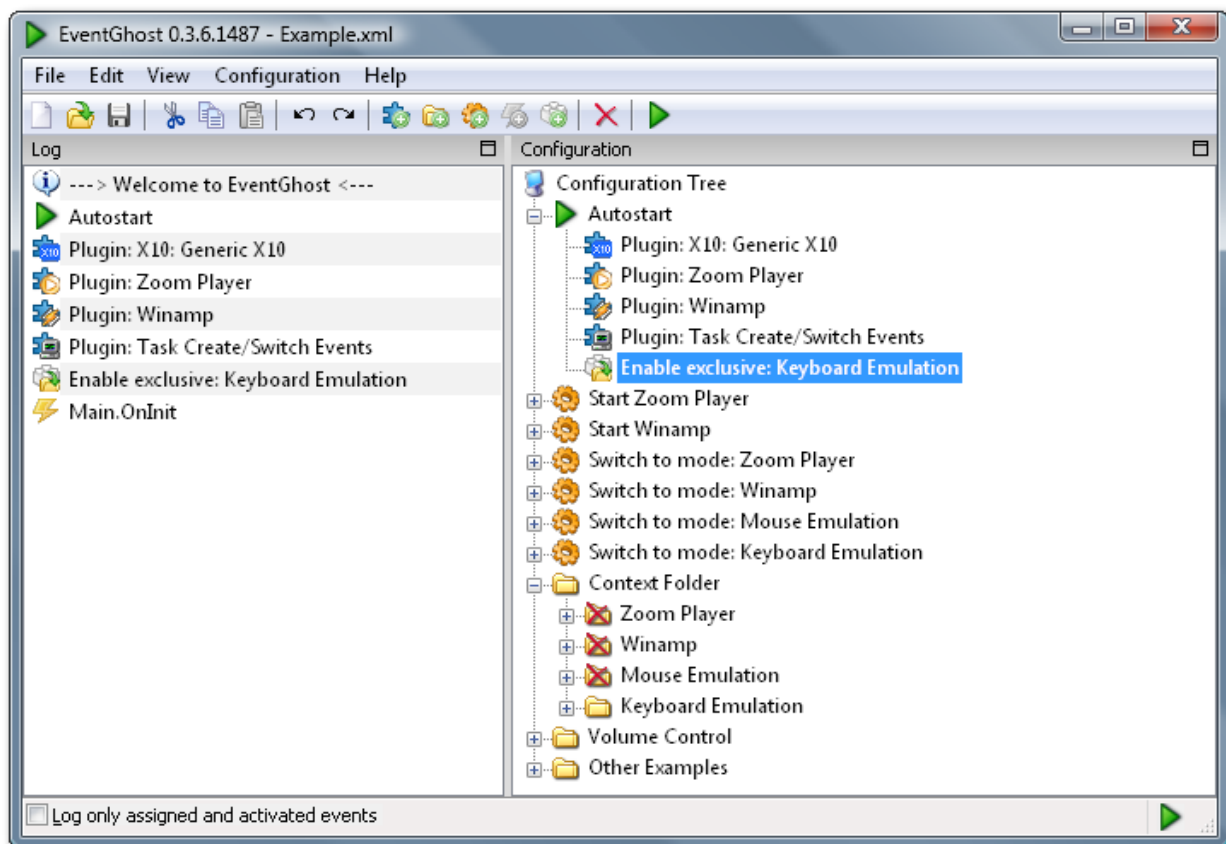
I consider the mouse function of EventGhost to be far superior to the Firefly's original software. The speed of the cursor slowly ramps up the longer the button is pressed. One of the most refreshing things about this program is that it uses about 8 MB of RAM as compared to Snapstream's 108 MB. This significantly improved the reaction time of this computer. The USB-UIRT worked as a perfect substitute for an X10 remote. The IR transmitting also worked perfectly. I highly suggest the stick-on IR emitters I purchased from SMARTHOME. Both IR emitters were able to transmit perfectly to the VCR. You do not have to position your USB-UIRT directly in front of the device you wish to control this way. This also leaves the USB-UIRT to be positioned perfectly as a receiver. I have not found any pitfalls in the software besides having to restart the program once the USB-UIRT has been connected.

Conclusion

I was able to adjust the software to my satisfaction. This did take a few hours and a bit of trial and error to get the hardware working correctly. I found this to be about the same as programming a universal remote once you got the hang of it but at least this way you don't lose your settings when the batteries are pulled out. The main downfall of this software is that the users do not openly share the xml files that they have created. Everything must be done by you excluding the examples put in the program by default. The forum is extremely helpful and prompt with answers. They are also adding new hardware support and plugins constantly. I found the wiki created for this program to be a very poor source of information for beginners but considering the usefulness of this program I was able to work my way through it. This is intended to be a gateway article for those looking to enter the world of remote PC control and the sky is the limit as to where you can go from here. As usual in the free software HTPC world you have to become your own hardware expert to pull off what you're striving to do. This program cannot go beyond what the hardware is capable of.

On a side note, I highly suggest *donating or contributing* to this project to ensure its future success. If your device is not currently supported just check back at the *supported hardware* page from time to time to see what's new.

Main Window



Logger Pane

Configuration Pane



Configuration Tree

The Configuration Tree contains the complete EventGhost configuration and is made up of containers and items.

All containers and items except Autostart can be enabled/disabled manually and by Actions.



Autostart

Autostart is a special container that can hold Plugins to load and Actions to execute every time the Configuration Tree is loaded (eg EventGhost is started). There is always a single Autostart item in the Configuration Tree and it cannot be removed or disabled.



Folders

Folders are named containers that can hold Macros and/or more Folders. They are used to structure your configuration as you see fit.



Macros

Macros are named containers that can hold Events to trigger the Macro and Actions to execute when the Macro is triggered.

For example:

- You press the button labeled *TV* on your remote control.
- The Plugin for your remote sees that you pressed the *TV* button and generates a *TV* Event.
- Any active Macros that contain the *TV* Event are triggered and begin executing their Actions.



Plugins

Plugins extend the functionality of EventGhost by generating Events and/or adding additional Actions. Plugins are always placed in the Autostart container.



Events

Events are the triggers that cause Macros to execute their Actions. They are generated by EventGhost and Plugins. Events generated by Plugins are usually prefixed with the Plugin's name, so if you were using an X10 remote in the example above, the X10 Remote Plugin would have generated an X10.TV Event.

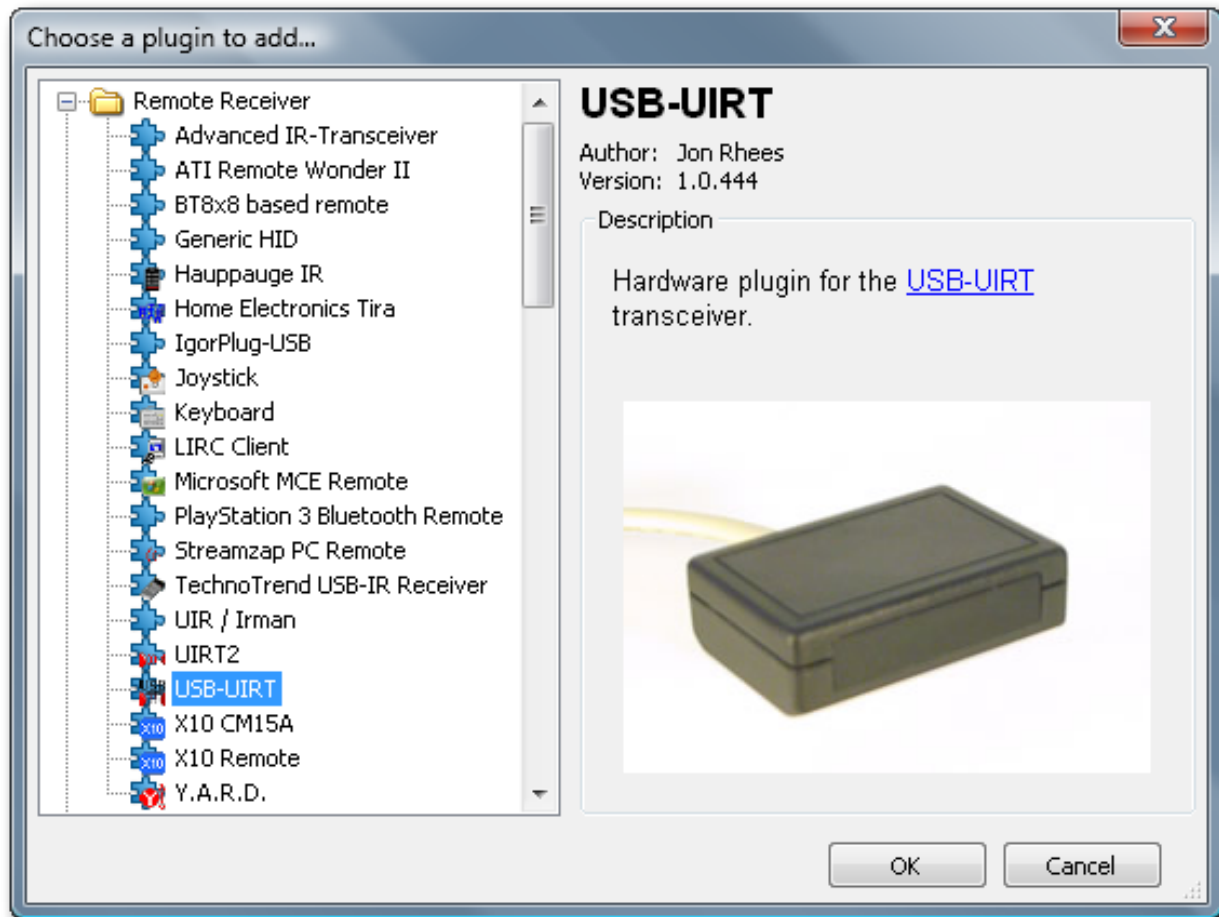
Actions

Actions are the commands that are executed when a Macro is triggered by an Event. They are provided by Plugins.

Adding Plugins

Adding new Plugins to your configuration is very simple:

- Select the *Configuration→Add Plugin...* menu item
- Select the Plugin you want to add from the list
- Click the OK button



The Plugin you selected will be added to the Autostart container and a dialog may appear to configure additional settings if needed.

See the [List of Plugins](#) for more details on the available Plugins.

Assigning Events to Macros

There are three ways you can assign trigger Events to a Macro:

1. Drag-and-drop an Event from the Log to an existing Macro in the Configuration Tree.
2. Select an Event in the Log and copy it to the clipboard using `Ctrl+C`, *Right-Click→Copy*, or *Edit→Copy*, then select an existing Macro in the Configuration Tree and paste the Event using `Ctrl+V`, *Right-Click→Paste*, or *Edit→Paste*.
3. Select an existing Macro in the Configuration Tree and then add a new Event using *Right-Click→Add Event*, *Configuration→Add Event* or clicking the *Add Event* toolbar button.

You can rename any existing Event by selecting it and pressing `F2`, or selecting *Right-Click→Rename Item* or *Configuration→Rename Item*, and typing the new Event name.

Other common things that can act as a trigger for an event are:

- key presses (hotkeys)
- joysticks/gamepads
- a program is starting or is switched to by the user
- another program like EventGhost (on another PC), Girder or NetRemote is sending an event through TCP/IP
- a special HTTP request is made to the internal web server
- another program is sending an event through ActiveX

and everything some code can catch, as events can also be generated through plugins.

EventGhost gives the user a GUI to configure macros that do all kind of things like:

- launching applications
- emulating keystrokes
- emulating mouse movements and clicks
- control the sound card
- move, resize, etc. windows on the desktop
- execute Python scripts (Python interpreter and editor is built-in)
- transmit IR-codes to external consumer equipment, if you have a supported IR-transceiver.
- control external hardware devices like projectors and other media equipment through RS232 communication
- extensive control of programs that have special communication interfaces, such as some media players

and everything some code can do, as these list of actions can also be extended through plugins.

You can take a look at the list of plugins to find out what has been implemented also.

The plugin system is the most integral part of the program. Actually EventGhost is designed around the plugin idea from the beginning. Every action EventGhost does and every event it sees is implemented through a plugin, even the

most basic ones. So every plugin has rights equal to the built-in functions, because they are actually the same. The user can configure and use them through a consistent and, hopefully, easy to learn interface.

EventGhost is written mostly in Python with some low-level parts in C. Plugins can be written in any language that can produce DLLs, such as C, C++, Delphi and Visual Basic. But of course they can be (and mostly are) written in Python.

Supported Remote Receivers

Already supported devices

- **HID (Human Interface Device) based devices:**
 - Logitech OEM UltraX Media Remote
 - Gyration Media Center Remote
 - And surely a lot more receivers and devices.
- **X10-RF-Receivers that are normally bundled with remotes like:**
 - ATI Remote Wonder
 - ATI Remote Wonder PLUS
 - SnapStream Firefly
 - NVIDIA Personal Cinema Remote
 - Marmitek PC Control
 - Pearl Q-Sonic Master Remote 6in1
 - Niveus PC Remote Control
 - Medion RF Remote Control
 - Packard Bell RF MCE Remote Control OR32E
- USB-UIRT (transmitting IR is also supported)
- Y.A.R.D. (transmitting IR is also supported)
- IgorPlug-USB
- Streamzap PC Remote
- UIRT2 (transmitting IR currently is only partly supported)
- Microsoft MCE Remote (receiving only, transmitting IR will be supported in the future)

- Tira2 (transmitting IR is also supported)
- Egon USB receiver
- TechnoTrend USB-IR Receiver
- UIR
- Irman
- ATI Remote Wonder II
- Sony PlayStation 3 Bluetooth Remote

Not directly supported devices

- LIRC Serial Port Receiver
- IgorPlug

You can read the reason why these devices currently can't be directly supported in EventGhost [here](#). But you can use them in conjunction with [WinLIRC](#) and the plugin that is [discussed here](#).

If your device is not listed here, feel free to ask on the [EventGhost Support Forum](#) if a plugin can be written for it.

List of Plugins

This is the list of the 113 plugins currently distributed with EventGhost WIP:

Essential (always loaded)

EventGhost Actions to control events, macro flow, and the configuration tree.

Mouse Actions to control the mouse cursor and emulation of mouse events.

System Actions to control various aspects of your system, including audio, display, power, and registry.

Window Actions to control windows on your desktop, like finding specific windows, moving, resizing, and sending keypresses to them.

Remote Receiver

ASUS PSR-2000 A plugin for the ASUS PSR-2000 remote.

ATI Remote Wonder II (WinUSB) Plugin for the ATI Remote Wonder II remote.

BT8x8-based Remote AverTvStudio remote, may work for other bt8x8 cards. Requires DScaler to be installed.

Conceptronic CLLRCMCE Plugin for the Conceptronic Remote Control for Windows® Media Center.

CyberLink Universal Remote Control Plugin for the CyberLink Universal Remote Control

ELV FS20 PCE Allows to receive events from FS20 remote controls.

EventPhone Remote Plugin for the EventPhone iPhone/Ipod Touch™ native application.

Generic HID Communication with devices that follow the Human Interface Device (HID) standard.

Hauppauge IR Hardware plugin for the Hauppauge IR Control, delivered with several Hauppauge TV cards

Home Electronics Tira Hardware plugin for the Home Electronics Tira transceiver.

IgorPlug-USB Plugin for [Igor Cesko's USB IR](#) receiver.

Joystick Use joysticks and gamepads as input devices for EventGhost.

Keene IR Anywhere Hardware plugin for the [Keene Electronics IR Anywhere](#) transceiver

Keyboard This plugin generates events on keypresses (hotkeys).


LIRC Client Plugin for sending and receiving LIRC eventstrings. Generates EventGhost events based on data received from the LIRC-server.

Logitech UltraX Media Remote Plugin for the Logitech UltraX Media Remote.

Microsoft MCE Remote Plugin for the Microsoft MCE remote.

Microsoft MCE Remote (Vista+) Plugin for the Microsoft MCE remote. Requires installation of AlternateM-ceIrService.

PC Remote Controller Plugin for the PC Remote Controller.

PHX01RN 

PlayStation 3 Bluetooth Remote Hardware plugin for the PS3 Bluetooth Remote (based on the HID code of Bartman)

Sidewinder Game Voice Allows the communication with the Microsoft Sidewinder Game Voice. This plug in also demonstrates how to use EventGhost's HID API.

Speed-Link SL-6399 Media Remote Plugin for the Speed-Link Media Remote Control (SL-6399)

Streamzap PC Remote Hardware plugin for the [Streamzap PC Remote](#).

TechniSat USB IR Receiver Plugin for the TechniSat USB IR Receiver

TechnoTrend IR Devices (BDA) Hardware plugin for TechnoTrend IR receivers delivered with TechnoTrend TV cards.

TechnoTrend USB IR Receiver Hardware plugin for the [TechnoTrend USB Infrared Receiver](#).

Terratec USB Receiver Plugin for the Terratec USB Remote

UIR / Irman Hardware plugin for the [Universal Infrared Receiver V1 \(UIR\)](#) and the [Evation.com Irman](#) device.

UIRT2 Hardware plugin for the [Universal InfraRed Transceiver V2](#).

USB-UIRT Hardware plugin for the [USB-UIRT](#) transceiver.

X10 Remote Hardware plugin for X10 compatible RF remotes.

Xbox Remote Control Xbox remote control plugin, based on the Generic Human Interface Device (HID) plugin.

Y.A.R.D. Hardware plugin for the [Y.A.R.D.](#) IR-transceiver from Andre Weber.

auvisio VRC-1100 Ro Plugin for the Auvisio PC-Remote.

Program Control

BSPlayer Adds support functions to control BSPlayer.

Billy Player Adds actions to control the [Billy](#) audio player.

CyberLink PowerDVD Adds actions to control CyberLink PowerDVD 7 and 8.

DScaler 4 Adds support functions for [DScaler 4](#).

DVB Dream Adds actions to control [DVB Dream](#).

DVBViewer Adds support functions to control DVBViewer Pro/GE and DVBViewerService and returns events.

GOM Player Adds support functions to control GOM Player.

IrfanView Adds actions to control IrfanView.

MCE Adds actions to control Windows Media Center.

Media Player Classic Adds actions to control
'Media Player Classic - Home Cinema '.

MediaMonkey Adds support functions to control MediaMonkey.

MediaPortal Adds actions to control MediaPortal.

Meedio Adds actions to control Meedio.

MyTheatre Adds actions to control the MyTheatre multimedia application.

RadioSure Adds actions to control the Radio?Sure!

SageTV Adds actions to control the SageTV Media Center.

Sceneo TVcentral Adds actions to control Sceneo TVcentral.

Screamer Radio Adds actions to control the Screamer radio.

Splash Lite Adds actions to control the Splash Lite - next generation player.

TheaterTek Adds actions to control TheaterTek.

VLC media player Adds actions to control the VLC media player.

Weather Uses the Google web service to retrieve current and forecasted weather information. More Info on <http://code.google.com/p/python-weather-api/>

Winamp Adds actions to control Winamp.

Windows Media Player Adds actions to control the Windows Media Player.

XBMC Event Receiver Receives events from XBMC Host UDP Broadcasts.

XBMC2 Adds actions buttons to control XBMC.

Zoom Player Adds actions to control the famous Zoom Player.

ffdshow Adds actions to control the ffdshow DirectShow filter.

foobar2000 Adds actions to control the Foobar2000 audio player.

For v1.0 you need I foosion's Run Command (foo_runcmd) plugin.

External Hardware Equipment

Barco CRT Projector Controls Barco CRT projectors via RS232.

CM11A Control a CM11A device via the serial interface

CambridgeAudio Amps Serial Control Cambridge Audio Amps via RS232

Denon AV Serial Control Denon A/V Amps/Receivers via RS232

ELV FS20 PCS Allows to send commands to FS20 receivers.

FHZ 1000 PC FHZ 1000 PC

Homeseer Homeseer plugin. More info on <http://smart-living.geoblog.be/>

JVC DILA Projector This plug in is for controlling JVC DILA projectors via RS-232

JVC HD-1 Projector This plugin is for controlling an JVC HD-1 projector via RS-232

Marantz AV Receiver This plugin allows you to control your **Marantz** SR-series receiver through it's serial port.

Onkyo AV Serial Control Onkyo A/V Receivers via RS232

Onkyo ISCP Controls any Onkyo Receiver which supports the ISCP protocol.

Optoma H79 Serial Control an Optoma H79 projector via RS232

RFXcom_xPL Sends and receives RFXCOM messages using xPLRFX from xPL Monkey.

<center></center>

Samsung TV Control Samsung TV via RS232

TellStick Plugin to control TellStick devices.

Yamaha RX-V1000 Serial Control Yamaha RX-V1000 receivers using RS232.

d-box2 Remote Emulator Control your d-box2 set-top box over Ethernet.

xPL Send and receive xPL messages.

Other

Broadcaster Listens for and Transmits UDP Broadcasts

Desktop Remote Creates a remote-like desktop window.

Directory Watcher Monitors a directory and generates events if files are created, deleted or changed in it.

Dynamic Webserver Implements a small webserver, that you can use to generate events through HTML-pages.

File Operations File Operations (reading, periodical reading and writing).

Multitap Adds Multitapper actions.

Network Event Receiver Receives events from Network Event Sender plugins.

Network Event Sender Sends events to an Network Event Receiver plugin through TCP/IP.

OS Menu Allows you to create custom On Screen Menu.

On Screen Explorer Allows you to create custom On Screen Explorer.

Ping This plugin generates events when an host become available or unavailable on your LAN. It uses the ping commands of windows. Please, have a look at the readme file !

Process Watcher Generates events if a process is created or destroyed

Pushover This plugin sends notifications to you mobile using the Pushover service. For more details visit: <https://pushover.net/>

Remote Event Mapper This plugin is designed for easy remapping of events from remote controls.


ScheduleGhost This plugin is designed to easily schedule events ...

Scheduler Triggers an event at configurable dates & times

Serial Port Arbitrary communication through a serial port.

Sound Mixer Ex This plugin allows you to set virtually any control available on your soundcard.

Speech Uses the Text-To-Speech service of the Microsoft Speech API (SAPI).

SunTracker Triggers an event at sunset/sunrise and configurable dates & times


System Tray Menu Allows you to add custom menu entries to the tray menu of EventGhost.

Task Monitor Generates events when an application starts, exits, flashes the taskbar, or gets switched into focus.

Test Pattern Plugin to show some test patterns.

Timer Triggers an event after an adjustable time and optionally repeats it after an interval.

USB/RFID-Interface Receives events from ELV USB/RFID-Interface.

Webserver Implements a small webserver, that you can use to generate events through HTML-pages and WebSocket.

Scripting with Python

Introduction

If you want to script with Python in EventGhost you first have to understand the difference between the environment of a ‘PythonScript’ and a ‘PythonCommand’ action in EventGhost.

PythonCommand

PythonCommands execute a single-line Python statement or expression. All PythonCommands share a single global namespace, so you can create a global variable with one PythonCommand and directly modify this variable with another PythonCommand later. The global namespace includes all Python built-in objects and the special object `eg`, that we will explain later.

PythonScript

Every PythonScript in EventGhost has its own global namespace. The global namespace includes all Python built-in objects and the special object `eg`.

The one and all ‘eg’ object

Everything special that is needed from EventGhost for scripting and writing plugins is stuffed into the `eg` object. It includes many functions, variables, classes and objects. You could actually say `eg` is EventGhost itself.

So we will explain some of the members of `eg` here in more detail:

Useful members

eg.globals

As explained before, PythonCommands all share a single global namespace. This is actually `eg.globals`. Since every PythonScript has its own global namespace, you can't directly access a value defined with a PythonCommand from a PythonScript and vice versa, but you can access it through `eg.globals`.

If you made, for example, a PythonCommand like `'myVar = 123'`, you can later write in a PythonScript:

```
print eg.globals.myVar
```

and you will get the value `'123'` printed to the logger. You can of course also create new named variables in this namespace from a PythonScript by simply writing:

```
eg.globals.myOtherVar = "Hello World!"
```

and after that use the PythonCommand `'print myOtherVar'` to get the value.

So `eg.globals` is the bridge between values defined with PythonCommands and values defined with PythonScripts and can also be used to transfer data between different PythonScripts.

eg.plugins

One neat feature of EventGhost is the ability to also use nearly every action of a plugin in a PythonScript. Every plugin creates a named member in `eg.plugins`, when it is loaded. And every action is a named member of a plugin. To find out the name of the action (and its plugin) you can simply create an action in EventGhost's tree, copy this action-item and paste it to a text editor like Notepad (or even EventGhost's built-in PythonScript editor). As an example, you will see something like this for `'System/Turn Mute Off'`:

```
<?xml version="1.0" encoding="UTF-8" ?>
<EventGhost Version="532">
  <Action>
    System.MuteOff()
  </Action>
</EventGhost>
```

If you look at the line between the `<Action>` tags, you will see the plugin is named `'System'` and the action is named `'MuteOff'`. Now you can write in a PythonScript:

```
eg.plugins.System.MuteOff()
```

and it will exactly do the same as a `'System/Turn Mute Off'` action.

Actions might also have parameters, so you will find that a `'Window/Resize'` action, that should resize a window to 200 pixels width and 300 pixels height would be called from a PythonScript as:

```
eg.plugins.Window.Resize(200, 300)
```

by simply configuring the action before you copy it and looking at the copied XML chunk in an editor.

eg.event

`eg.event` represents the event that is currently processed. Since your PythonScript or PythonCommand is most likely triggered by an event when it is executing, `eg.event` will give you information about this event. Most useful members of this object are:

eg.event.string This is the full qualified event string as you see it inside the logger, with the exception that if the `eg.event.payload` field (that is explained below) is not `None` the logger will also show it behind the event string, but this is not a part of the event string we are talking about here.

eg.event.payload A plugin might publish additional data related to this event. Through `eg.event.payload` you can access this data. For example the ‘Network Event Receiver’ plugin returns also the IP of the client that has generated the event. If there is no data, this field is `None`.

eg.event.prefix This is the first part of the event string till the first dot. This normally identifies the source of the event as a short string.

eg.event.suffix This is the part of the event string behind the first dot. So you could say:

```
eg.event.string = eg.event.prefix + '.' + eg.event.suffix
```

eg.event.time The time the event was generated as a floating point number in seconds (as returned by the `clock()` function of Python’s `time` module). Since most events are processed very quickly, this is most likely nearly the current time. But in some situations it might be more clever to use this time, instead of the current time, since even small differences might matter (for example if you want to determine a double-press).

eg.event.source This is the object that has generated the event. For most events, the source is a plugin-object but for some built-in events and events generated through `eg.TriggerEvent()`, this will be our beloved `eg` object.

eg.event.isEnded This boolean value indicates if the event is an enduring event and is still active. Some plugins (e.g. most of the remote receiver plugins) indicate if a button is pressed longer. As long as the button is pressed, this flag is ‘False’ and in the moment the user releases the button the flag turns to ‘True’. So you can poll this flag to see if the button is still pressed.

eg.result

Every action in EventGhost returns a result. For most actions this is simply Python’s `None`, but some might return a result that is useful for later evaluation. For example, the ‘Window/Find Window’ action returns a list of the window-handles it has found (or an empty list if it hasn’t found anything). So you can place a `PythonScript` directly after the ‘Find Window’ action and do something with this list.

The ‘EventGhost/Jump’ action also uses `eg.result` as the condition to decide what it has to do. If `eg.result` is determined as `True` by Python’s standard truth testing procedure, the ‘Jump’ action will regard the result of the last action as ‘successful’ and do a jump if configured so. So you can use this circumstance to control a ‘Jump’ from a `PythonCommand` or `PythonScript`, by assigning something to `eg.result`. For a `PythonCommand` you actually don’t need to assign directly to `eg.result`, because the result of a Python evaluation is automatically assigned to `eg.result`. If you make a `PythonCommand` like `myVar == 1`, EventGhost will compute this to `True` if ‘myVar’ is 1 or to `False` if ‘myVar’ is anything other and assign this `True/False` result to `eg.result`.

Useful functions

eg.TriggerEvent (*eventstring*)

To generate a new event in a `PythonScript`, you can use this function. Example usage:

```
eg.TriggerEvent("MyEvent")
```

This will generate a “Main.MyEvent” event. Actually you could also use the ‘EventGhost/TriggerEvent’ action with `eg.plugins.EventGhost.TriggerEvent("MyEvent")`, following the pattern described above, but for convenience this function is also exposed directly from `eg`.

eg.Exit ()

Sometimes you want to quickly exit a PythonScript, because you don't want to build deeply nested if-structures for example. `eg.Exit()` will exit your PythonScript immediately.

eg. **StopMacro** ()

Instructs EventGhost to stop executing the current macro after the current action (thus the PythonScript or Python-Command) has finished.

Contents

- [FAQ](#)
 - *Q: Isn't the installer huge for such a tool?*
 - *Q: Can I use an IrDA dongle to control my PC with a remote?*
 - *Q: How can I install EventGhost on a Windows 2000 machine?*

Q: Isn't the installer huge for such a tool?

A: Yes, it is. But you don't need to be afraid that the program will load all this into the memory. The installer includes a complete Python and wxPython runtime with all standard libraries, even if the program and its plugins don't use every aspect of them. This way everybody can use the rich set of Python modules for scripting and writing plugins.

Q: Can I use an IrDA dongle to control my PC with a remote?

A: No. [CIR \(consumer IR\)](#) that is used by your TV remote for example, is a complete other thing than [IrDA](#). They use different frequencies, encodings and modulation. IrDA is simply not made for the purposes of CIR and vice versa. However, there are some IrDA dongles that are advertised to work with CIR. One that I know of is the ACTiSYS IR200L. But other projects have found that this device won't work very reliably and therefore the EventGhost project has made no effort to support it.

Q: How can I install EventGhost on a Windows 2000 machine?

To run EventGhost you need GDI+ installed on your machine. GDI+ is a new graphics technology developed by Microsoft. It provides rich set of additional graphics features for 3rd-party developers. GDI+ is distributed as part of Windows XP and above, while for other operating systems an additional redistributable file installation is required. Since this file would increase the size of the EventGhost installer by one megabyte and Windows 2000 is not that much used any more, the needed GDIPLUS.dll is not included in the installer.

To install EventGhost on a Windows 2000 machine please follow the steps mentioned in this Microsoft Knowledge Base article: <http://support.microsoft.com/kb/915052>

After you have installed GDIPLUS.dll, you should be able to install and run EventGhost.

Command Line Options

The EventGhost main executable accepts the following command line arguments:

Note: If you started EventGhost with elevated privileges you must also run all further commands with elevated privileges.

-event <eventname> [<payload> ...]

Issues the event <eventname> in the currently running EventGhost instance. Optionally you can specify one or more <payload> strings, that will be added to the event in the `eg.event.payload` field.

-e <eventname> [<payload> ...]

Shorter alias for the `-event` option.

-hide

Starts EventGhost hidden in the system tray. Otherwise it would start in the state it had when it was closed.

-h

Shorter alias for the `-hide` option.

-file <filename>

Opens the XML configuration file <filename> on start-up (instead of the last loaded file).

-f <filename>

Shorter alias for the `-file` option.

-n <host>:<port> <password> <eventname> [<payload> ...]

This one is similar to the `-event` option, but sends the event <eventname> through TCP/IP like the 'Network Event Sender' plugin does. It will not start EventGhost, so it can be used as a little helper tool for other applications or .BAT files to send events to a remote machine. <host> has to be the IP or host name of the target machine. <port> and <password> are the options that you have configured on the target machine's 'Network Event Receiver' plugin.

-translate

Starts EventGhost's translation editor.

-configdir <directory>

Instructs EventGhost to use the directory <directory> to store and retrieve its settings. Without this option EventGhost uses a directory in the application data folder of your machine for storing its settings. For example, through this option you can change the folder to a location on a USB stick to make EventGhost portable.

CHAPTER 9

Contributing

Contributions to the project are always welcome. You can do it in many different ways:

- Write or enhance the documentation for EventGhost.
- Write plugins for EventGhost
- Translate EventGhost to your language.
- Help other people on the EventGhost Forum.
- Donate money to this project through PayPal.

Financial donations will only be used for EventGhost related purposes, like paying the bills for the website or to add more and better support for different hardware.

CHAPTER 10

X10 Remotes

If you notice that your remote sends keystrokes even if EventGhost is not started, you most likely have installed the wrong driver. Some manufacturers deliver a driver that registers the remote as a HID (Human Interface Device). Even though HID might be fine if you really don't want to use any third party program, it gives you trouble with tools like EventGhost because they are not able to stop these drivers from sending keystrokes.

To make sure which driver your remote currently uses, open Device Manager and look for a category "Human Interface Devices". If you find an entry named "X10 Hid Device" there, you have the wrong driver.



Sadly this X10-HID driver is not so easy to uninstall. If you right click on the entry, you won't find any option to uninstall it. To get rid of it, you have to use some tricks:

(These instructions are for Windows XP)

1. Unplug the USB receiver from your system.
2. Right click on the "X10 Hid Device" entry and choose *Update Driver*.
3. The "Hardware Update Wizard" will popup. If you are asked to connect to Windows Update, choose the *No, not this time* radio box and press *Next*.

4. On the next page you are asked what you want the wizard to do. Choose *Install from a list or specific location (Advanced)* and press *Next*.
5. On the next page select *Don't search, I will choose the driver to install.* and press *Next*.
6. Now you should see a list that includes the “X10 Hid Device” entry. Above this list is a check box called *Show compatible hardware*. Un-check this box.
7. On the left side you now see a manufacturer list. Scroll it up to the top and select *(Standard system devices)*.
8. The first entry on the right side should now be *2-axis, 2-button joystick*. Select this entry and press *Next*.
9. You will now get a warning dialog that it is not recommended to install this driver. Ignore it and press *Yes*.
10. Now press *Finish* on the last page. If you are asked to restart your computer now, affirm it with *Yes* and let the computer reboot.
11. After your computer is back, open Device Manager again. You will now see the joystick entry with a yellow exclamation mark. Right click this entry to open the context menu and choose *Uninstall*. Affirm that you really want to uninstall it on the next dialog and the entry should disappear.

Now you should install the proper driver before you reconnect your receiver.

You can download the driver here:

- [X10 driver for 32-bit Windows](#).
- [X10 driver for 64-bit Windows](#).

CHAPTER 11

Indices and tables

- `genindex`
- `search`

Symbols

-configdir <directory>
 command line option, 33

-e <eventname> [<payload> ...]
 command line option, 33

-event <eventname> [<payload> ...]
 command line option, 33

-f <filename>
 command line option, 33

-file <filename>
 command line option, 33

-h
 command line option, 33

-hide
 command line option, 33

-n <host>:<port> <password> <eventname> [<payload>
 ...]
 command line option, 33

-translate
 command line option, 33

C

command line option

- configdir <directory>, 33
- e <eventname> [<payload> ...], 33
- event <eventname> [<payload> ...], 33
- f <filename>, 33
- file <filename>, 33
- h, 33
- hide, 33
- n <host>:<port> <password> <eventname> [<pay-
 load> ...], 33
- translate, 33

E

eg.Exit() (built-in function), 29

eg.StopMacro() (built-in function), 30

eg.TriggerEvent() (built-in function), 29