SEMI-INTELLIGENT SCREENSAVER

Couple days ago I described couple of interesting ways to interact with KDE using D-Bus message bus. Today I will show how to set up semi-intelligent screensaver.

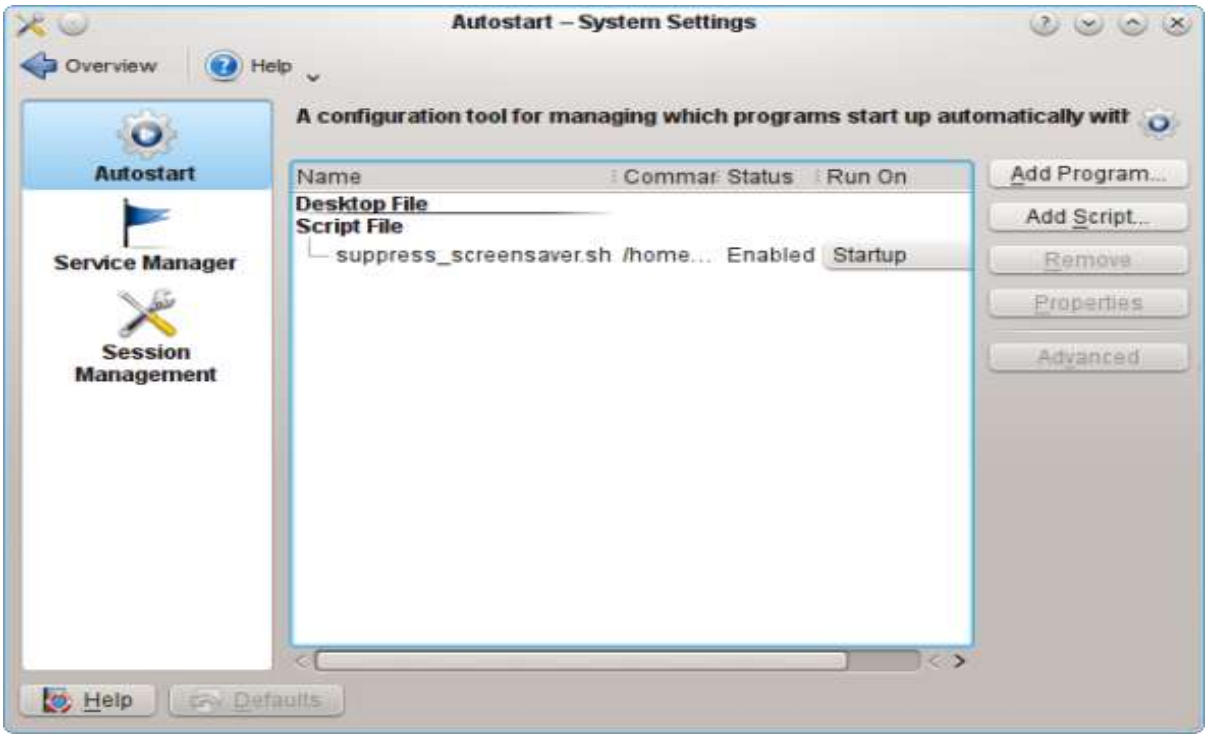
Let's start and create simple shell script

This simple shell script will simulate user activity if session idle time is greater then 50 seconds and Google Chrome use more then 5% CPU. It will prevent screensaver from kicking.

```
#!/bin/sh
# Simple script to demonstrate D-Bus usage

while true
do
  # read google chrome cpu usage
  ret=`top -b -n1 -u $(whoami) | awk '$12 ~ /chrome/ {SUM += $9} END {print SUM}'`

 if [ -n "$ret" ] && [ "$ret" -gt 5 ]; then
    idle_time=`qdbus org.kde.screensaver /ScreenSaver GetSessionIdleTime`
    if [ "$idle_time" -gt 50 ]; then
      qdbus org.kde.screensaver /ScreenSaver SimulateUserActivity
    fi
```

```
 fi


sleep 50

done
```

This script is designed to run in the background. The easiest way to execute it on KDE startup is to open *System Settings* > *Startup and Shutdown* > *Autostart* and add it there.



Let's look at previously created script from different way

You can try different approach and use xautolock utility as it provides a couple of nice features (very interesting possibilities to extend screensaver behavior).

To install *xautolock* execute command:

```
$ sudo apt-get install xautolock
```

Previously created script can be simplified because *xautolock* will monitor user activity.

```
#!/bin/sh
# Simple script to demonstrate xautolock usage

# read google chrome cpu usage
ret=`top -n1 | awk '$12 ~ /chrome/ {SUM += $9} END {print SUM}'`

if [ -n "$ret" ] && [ "$ret" -gt 5 ]; then
  qdbus org.kde.screensaver /ScreenSaver SimulateUserActivity
fi
```

To execute the above script use similar command:

```
$ xautolock -time 1 -locker /home/milosz/bin/suppress_screensaver_xautolock.sh
```

This utility is designed to activate screensaver but it doesn't mean that it cannot be used otherwise ;)

Let's extend the idea and write Ruby script

This is more advanced solution as it uses *Inhibit* and *UnInhibit* methods to suppress screensaver. Just remember that you don't need to call *UnInhibit* on exit as cookie will be dropped automatically.

```
#!/usr/bin/env ruby


require 'dbus'


session_bus  = DBus::SessionBus.instance
screen_saver = session_bus.service("org.freedesktop.ScreenSaver").object("/ScreenSaver");
screen_saver.introspect
screen_saver.default_iface="org.freedesktop.ScreenSaver";


chrome_check="top -n1 | awk \'$12 ~ /chrome/ {SUM += $9} END {print SUM}\'"
play_sound="play -q /usr/share/sounds/KDE-Window-All-Desktops-Not.ogg"
cookie = nil


loop do
  # read google chrome cpu usage
  ret=%x[#{chrome_check}]


  if ret.to_i > 5 then
    # google chrome cpu usage is greater than 5 so suspend screensaver
    if cookie == nil then
```

```
      cookie = screen_saver.Inhibit("google-chrome", "playing video").first
    end
  else
    # google chrome cpu usage is less than 6 so resume normal screensaver behaviour
    if cookie != nil
      screen_saver.UnInhibit(cookie)
      cookie = nil
      %x[#{play_sound}]
    end
  end

  # repeat loop every 1 minute
  sleep 60
end
```

Execute above script at the KDE session startup in the same way as the first script.

Notes

You can extend scripts mentioned here to perform different actions depending on the time of day or running applications.

For reason unknown to me *HasInhibit* method will return *false* if *power management* is enabled but it doesn't prevent ruby script from working.

```
$ qdbus org.freedesktop.PowerManagement /org/freedesktop/PowerManagement/Inhibit HasInhibit
```

Source: https://blog.sleeplessbeastie.eu/2013/03/02/semi-intelligent-screensaver/