

## Running SPx Applications as Services

Converting applications including SPx Server and SPx Fusion into background services

### Summary

Many users of Cambridge Pixel's software applications run them with their normal native UI. However, backend applications including SPx Server for tracking and radar video distribution, SPx Fusion for track combination and SPx Track Manager for format conversion can also be run in the background, using methods supported by the Windows or Linux platforms for running applications as services. This application note describes how to configure and deploy some key applications as services.

### Contents

Introduction .....	1
Web Interface .....	1
Disabling the native UI .....	2
Windows using Task Scheduler .....	3
Windows using Service Wrapper .....	6
Linux using /etc/rc.local .....	8
Linux using systemd .....	8
Linux using init.....	9

### Introduction

This application note addresses the configuration and use of SPx Server and SPx Fusion as background/service applications. The techniques described here are, however, applicable to any SPx software application that does not rely on the availability of a native graphical user interface.

### Web Interface

Many SPx applications support a Web-based interface. This can be very useful as a substitute for the native UI when the application is being run in the background. To ensure this interface is available, set the appropriate parameters in the application's `.rpi` configuration file (this is a text file which can be edited using Notepad, vi or equivalent).

```
WebServer.Available = 1
WebServer.Port = 8090
```

Each application uses a different default web server port to avoid conflict, and normally this should be left at its default setting.

A basic login facility is supported by the application's built-in web server, and this may be activated by setting further parameters in the configuration file.

```
WebServer.AuthType = 1  
WebServer.Username = user  
WebServer.Password = password
```

The desired username and password can be substituted in place of those shown above. When the web server is configured in this way, a simple login screen will be presented as shown below.



### Disabling the native UI

The next stage in the process is to suppress any native UI that the application would normally provide.

#### SPx Server under Windows

To start SPx Server running on Windows as a background process, first set the parameter in the configuration file `SPxServer.rpi` that determines what level of user interface is available.

```
Svr.GuiAvailable = -1
```

If you run SPx Server with this configuration and no further changes, it will have no visible presence either on the Windows desktop or in the system tray. It can be seen running under Background Processes in the task manager, but otherwise cannot be managed.

#### Other applications under Windows

Applications including SPx Fusion, SPx Track Manager and other applications and utilities can be run without a UI by setting one of the parameters below (the full name may vary between applications, so look for the `SystemTray` parameter):

```
Svr.SystemTray = 1
Application.SystemTray = 1
```

Applications started directly in this way will appear in the system tray.

### SPx Server under Linux

To run SPx Server under Linux without its native UI, just use the executable `spxserver_ng_64` (or `spxserver_ng` for 32-bit systems) instead of the normal executable.

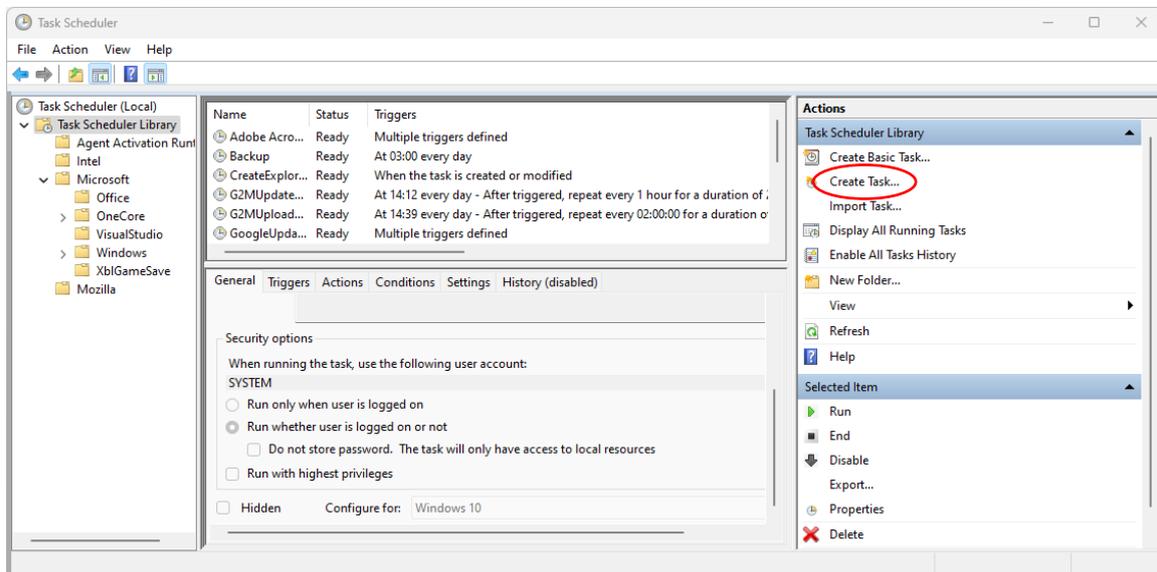
### Other applications under Linux

Other applications including SPx Fusion and SPx Track Manager only support a command-line interface under Linux, so the normal executable can be used.

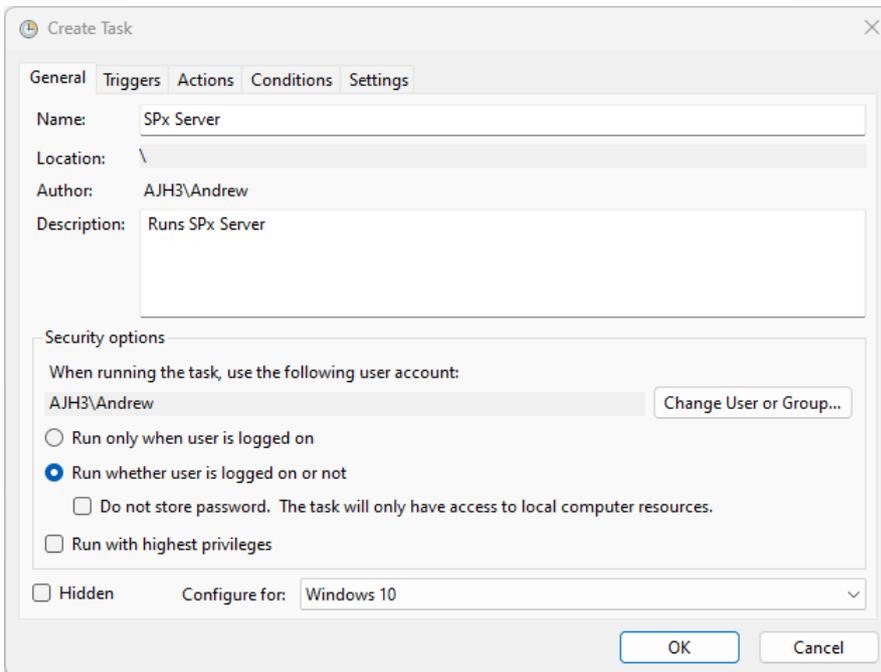
## Windows using Task Scheduler

The simplest method for running an application in the background under Windows without using any third-party software is to use the task scheduler. This allows an application to be run automatically at startup. The following instructions, while using SPx Server as an example, apply equally to any SPx application configured as above.

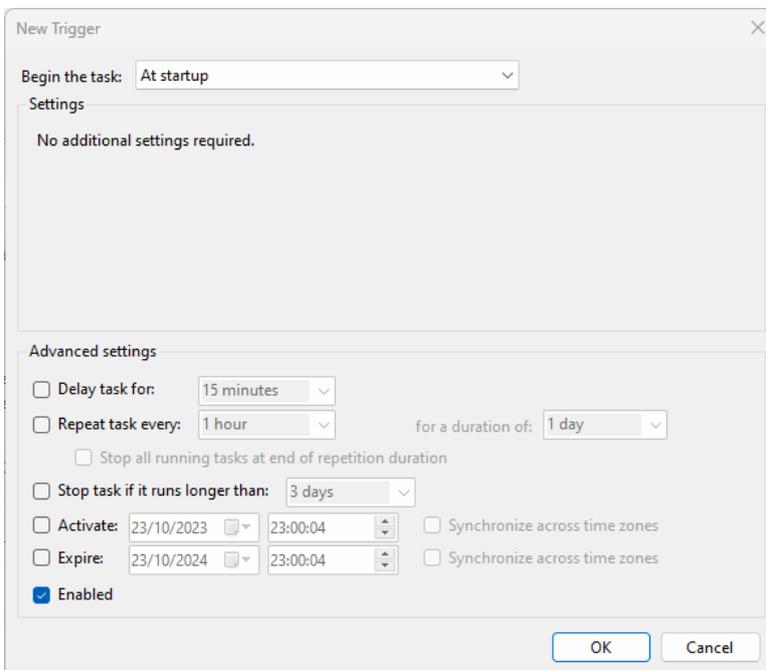
First, invoke the task scheduler from the Windows menu, and create a new task using the Create Task... option under Actions.



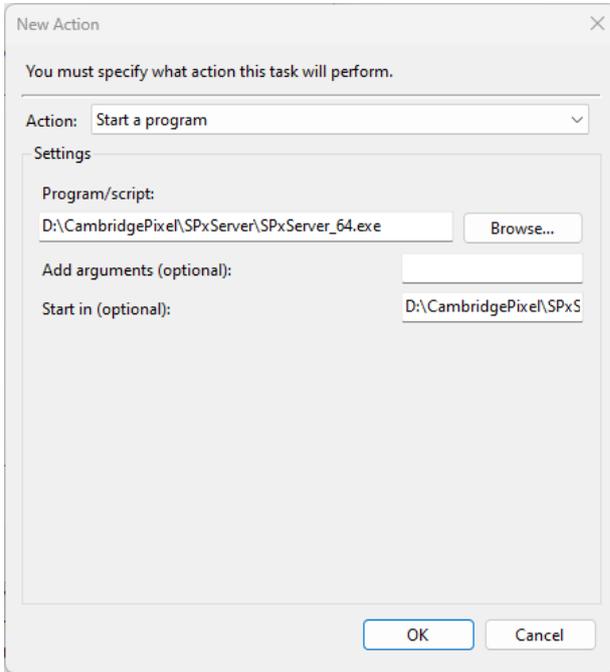
Populate the General tab with the task details:



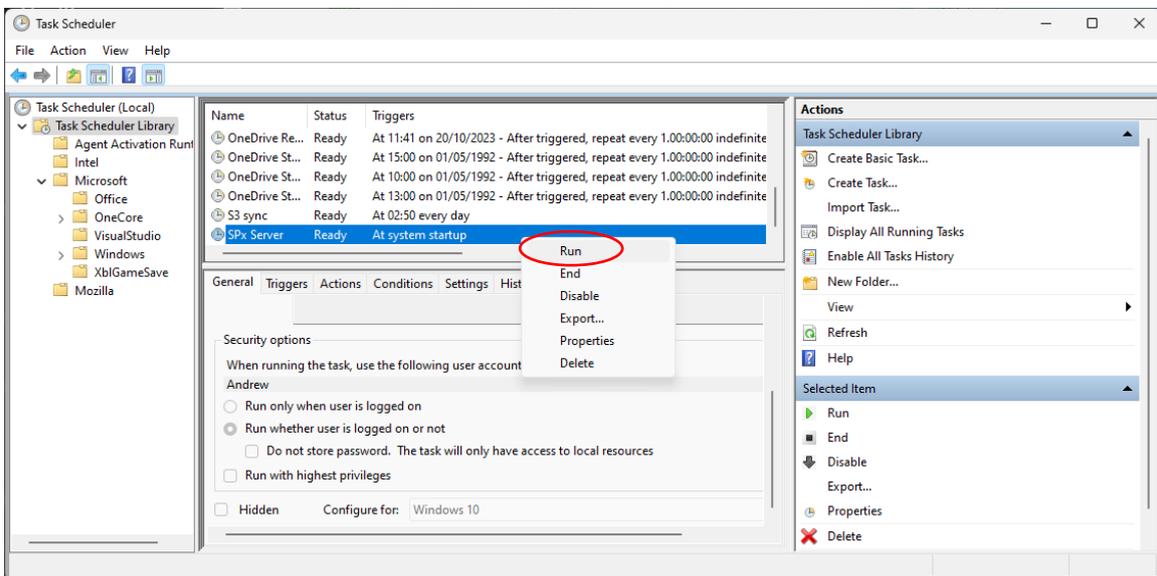
Use the New.. button in the Trigger tab to schedule the task to be run at system startup.



Then use the New.. button in the Actions tab to define the program to be run. Ensure you fill in the Start in field with the directory in which the SPx Server executable is located.



The setup can now be tested from the main task scheduler screen by right-clicking on the new task and selecting Run.



The running task will be visible under Background processes in Task Manager.

> Spooler SubSystem App	0%	2.3 MB	0 MB/s	0 Mbps	0%
> SPx Server	1.7%	239.4 MB	0 MB/s	0 Mbps	0%
> SQL Server VSS Writer - 64 Bit	0%	0 MB	0 MB/s	0 Mbps	0%

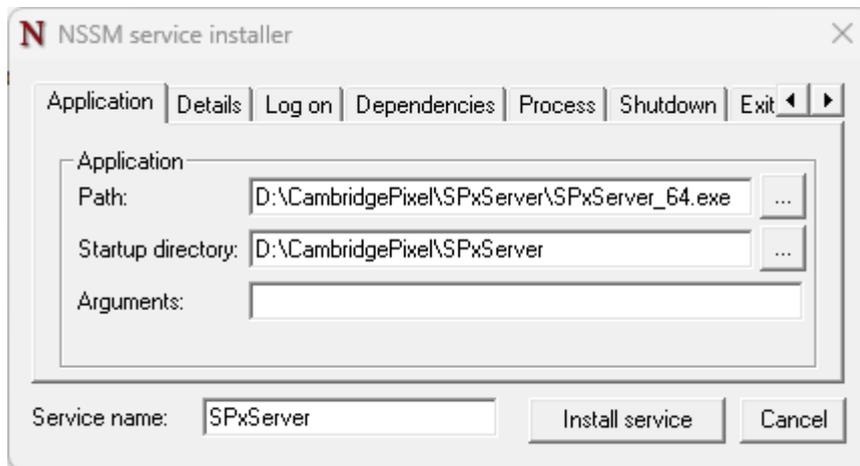
## Windows using Service Wrapper

A second approach involves using third-party software to convert the SPx application into a service using a wrapper. There are several options available, one of which is NSSM (<https://nssm.cc/>). This is mature and free software which nonetheless works well under Windows 10/11, and which will be used in this section to demonstrate the principle.

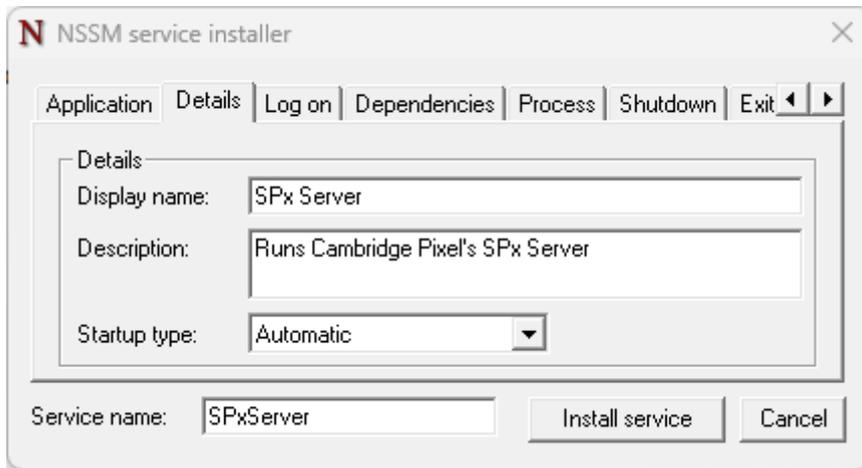
Once NSSM has been downloaded (use prelease build 2.24-101 or later), unpack it and place the executable from the **win32** or **win64** directory, as appropriate, into a directory referenced in your PATH environment variable. Then open a command prompt with Administrator privileges. (If a command prompt without elevated privileges is used, you will receive a Windows security prompt each time you run nssm). The following command should be used to create the service:

```
nssm install SPxServer
```

This will open an editing GUI into which the appropriate information can be entered into the relevant tabs.



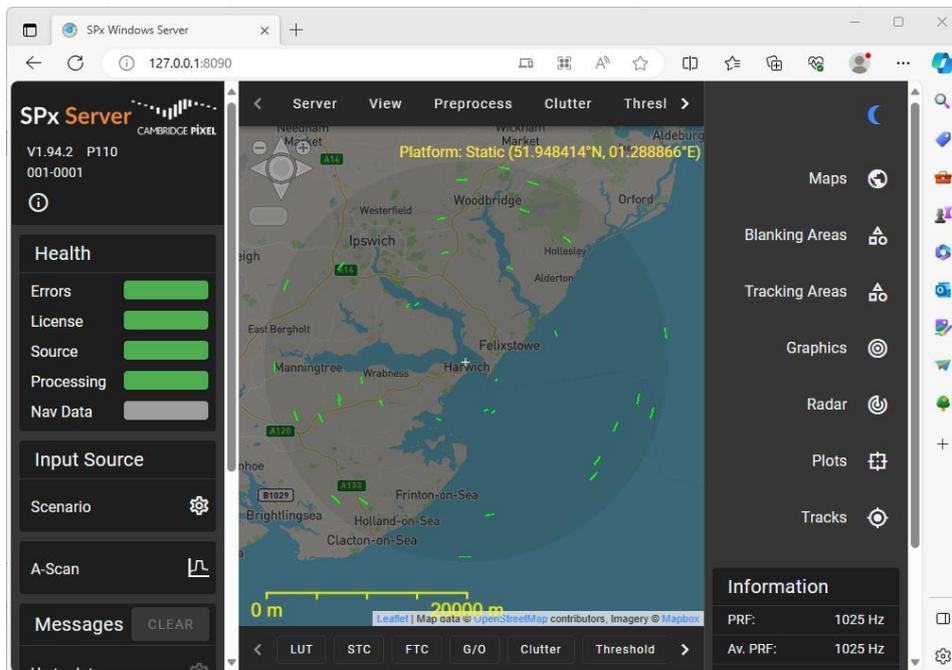
Note that if multiple instances of SPx Server are to be run, a service with a different name will need to be created for each one, specifying the appropriate configuration file in the Arguments field.



Once all fields have been completed, press the **Install service** button. You can then use the Services dialog to confirm that the service has been correctly installed.

Spatial Data Service	This service is used for Spatial Perception scenarios	Manual	Local Service
Spot Verifier	Verifies potential file system corruptions.	Manual (Trig...	Local System
SPx Server	Runs Cambridge Pixel's SPx Server	Automatic	Local System
SQL Server VSS Writer	Provides the interface to backup/restore Microsoft SQL server thr...	Running	Automatic
SSDP Discovery	Discovers networked devices and services that use the SSDP disco...	Running	Manual

Right-click on SPx Server and select Start to start the service manually, then access the web interface at <http://127.0.0.1:8090/> to confirm correct operation.



The service can be edited at any time using the command

```
nssm edit SPxServer
```

and can be removed using the command

```
nssm remove SPxServer
```

If using a MAC-locked licence file, ensure that it is located at the default location of `C:\cp-spx.lic` or in the executable directory of SPx Server.

### Linux using `/etc/rc.local`

Under Linux, the simplest method of automatically starting one or more SPx applications at startup is by the addition of commands to the file `/etc/rc.local`. For example:

```
cd /opt/CambridgePixel/SPxServer-V1.94/Servers/SPxServer
./spxserver_ng_64 &
```

If using a MAC-locked licence file, ensure that it is located at the default location of `/usr/local/SPx/cp-spx.lic` or in the executable directory of SPx Server. Otherwise, add a line

```
export SPX_LICENSE_FILE=/opt/CambridgePixel/cp-spx-001-0001
```

above the lines above, modifying the path and name of the licence file as appropriate.

### Linux using `systemd`

A more flexible and reliable approach is to use the **systemd** facility, widely supported in modern Linux distributions. This allows SPx applications to be started and stopped at will and to be automatically run at system startup. To use this approach, first create two scripts that will be used to start and stop SPx applications, and put them in a convenient location, making sure they have execute permission:

```
/opt/CambridgePixel/start-spx:
```

```
#!/bin/sh
exec 2>&1 >/opt/CambridgePixel/log.txt
export SPX_LICENSE_FILE=/opt/CambridgePixel/cp-spx-001-0001.lic
cd /opt/CambridgePixel/SPxServer-V1.94/Servers/SPxServer
exec ./spxserver_ng_64
```

```
/opt/CambridgePixel/stop-spx
```

```
#!/bin/sh
killall spxserver_ng_64
```

If using a MAC-locked licence file, ensure that it is located at the default location of `/usr/local/SPx/cp-spx.lic` or in the executable directory of SPx Server, or set its path explicitly as in the example above.

Next, create a system service file:

```
/etc/systemd/system/spx.service:  
[Unit]  
Description=SPx Server  
After=network.target  
  
[Service]  
Type=simple  
Restart=on-failure  
RestartSec=5  
ExecStart=/opt/CambridgePixel/start-spx  
ExecStop=/opt/CambridgePixel/stop-spx  
  
[Install]  
WantedBy=multi-user.target
```

The service can be enabled or disabled for automatic startup using these commands:

```
systemctl enable spx  
systemctl disable spx
```

The service can be manually started, stopped or restated using these commands:

```
systemctl start spx  
systemctl stop spx  
systemctl restart spx
```

If you need to run more than one SPx application, the scripts can be amended accordingly, ensuring that the last app to be started is not run in the background:

```
/opt/CambridgePixel/start-spx:  
#!/bin/sh  
exec 2>&1 >/opt/CambridgePixel/log.txt  
export SPX_LICENSE_FILE=/opt/CambridgePixel/cp-spx-001-0001.lic  
cd /opt/CambridgePixel/SPxFusion-V1.94.1/Servers/SPxFuse  
./spxfuse &  
sleep 2  
cd /opt/CambridgePixel/SPxServer-V1.94/Servers/SPxServer  
exec ./spxserver_ng_64
```

```
/opt/CambridgePixel/stop-spx:  
#!/bin/sh  
killall spxserver_ng_64  
killall spxfuse
```

### Linux using init

The **init** mechanism is the older mechanism for managing startup of applications. The systemd method should normally be used in preference. To use the init method, first create the `start-spx` and `stop-spx` scripts as above. Then create the following file:

```
/etc/init.d/spx:
#!/bin/sh
### BEGIN INIT INFO
# Provides:          spx
# Required-Start:    $local_fs $remote_fs $network $syslog
# Required-Stop:     $local_fs $remote_fs $network $syslog
# Default-Start:     2 3 4 5
# Default-Stop:      0 1 6
# Short-Description: Start spx
### END INIT INFO

PATH=/bin:/usr/bin:/sbin:/usr/sbin
DAEMON=spxserver_ng_64
. /lib/lsb/init-functions
case "$1" in
  start)
    log_daemon_msg "Starting SPx" "spx"
    start_daemon /opt/CambridgePixel/start-spx
    log_end_msg $?;;
  stop)
    log_daemon_msg "Stopping SPx" "spx"
    /opt/CambridgePixel/stop-spx
    log_end_msg $?;;
  status)
    if pidof $DAEMON >/dev/null 2>&1; then
      echo "$DAEMON is running"; exit 0
    else
      echo "$DAEMON is NOT running"; exit 3
    fi;;
  force-reload|restart)
    $0 stop
    $0 start;;
  *)
    echo "Usage: /etc/init.d/spx {start|stop|restart|force-reload}"
    exit 1
    ;;
esac
exit 0
```

The script can be enabled or disabled for startup using these commands:

```
update-rc.d spx defaults
update-rc.d spx remove
```

The applications can be manually started, stopped or restarted using these commands:

```
service spx start
service spx stop
service spx restart
```

< End of document >